

J. Morgenthale

**INGRES – ORACLE COMPARISON
HOW TO COMPETE**

**Prepared by Sandra Duerr
April, 1985**

This document is a comparison of INGRES and Oracle. It is meant as a training tool for RTI staff and is for **INTERNAL USE ONLY.**

It is NOT to be copied or given to prospects.

Periodic updates will be made available as additional information is obtained and after new versions of each system are released.

Table of Contents

1. INTRODUCTION		5
2. ORACLE OVERVIEW		6
3. HOW TO COMPETE AGAINST ORACLE		8
4. FEATURES ORACLE STRESSES TO WHICH WE MUST RESPOND		9
4.1 Null Values		9
4.2 Nested Queries		11
4.3 Portability		13
4.4 PC Product		14
4.5 SQL		16
4.6 Clustering		17
4.7 Add a Column to a Table		19
4.8 Multi-User Performance		20
4.9 Outerjoin		21
4.10 Tree Walks or Bill-of-Materials Queries		23
4.11 User Friendly Interface (UFI) Commands		24
4.12 Triggers		26
4.13 Record-Level Locking		27
4.14 Interactive Application Facility (IAF)		29
4.15 Multiple Retrieves in the Report Writer		32
4.16 Horizontal Scrolling		33
4.17 Soundex		34
4.18 Larger Record Length and Long Text Field		35
5. INGRES FEATURES WE SHOULD STRESS		37
5.1 Reliability/Security		37
5.2 Support		39
5.3 Product Development History		40
5.4 4th Generation Language (ABF/OSL)		41
5.5 Performance/Query Optimizer		42
5.6 Multiple Storage Structures		45

5.7 DECNET/Cluster Support	46
5.8 VIFRED	47
5.9 Multi-Table Query-By-Forms	48
5.10 Integrity Constraints	49
5.11 Space Management.....	50
5.12 EQUER/Forms.....	51
5.13 Preprocessors	52
5.14 Graphics.....	53
5.15 Locking/Concurrency Control	54
5.16 Load/Unload Utility.....	55
5.17 Integrated Data Dictionary.....	56
5.18 Recovery/Audit Trails.....	57
5.19 Default Forms, Reports, and Graphs	58
5.20 Transaction Management/Savepoints	59
5.21 Report-By-Forms/Report-Writer	60
5.22 QUEL	61
5.23 INGRES/Menu	62
5.24 Macro Utility	63
5.25 Price.....	64

1. INTRODUCTION

This report presents a comprehensive description of one of our competitors, Oracle, a relational database management system offered by Oracle Corp. This report describes the following items:

- The major parts of Oracle and the INGRES counterparts to which they should be compared
- The major strengths as seen by Oracle (i.e., the features which they stress in selling against us) and how to respond
- INGRES's strong points, which should be used in selling against Oracle.

This report is complemented by the *Feature Analysis of INGRES and Oracle*, which is a detailed, feature-by-feature comparison.

The information for this report was obtained from various sources (e.g., Oracle users, ex-Oracle employees, and Oracle documentation) and has been verified when possible. Any assumptions made are clearly indicated. INGRES Version 3.0 has been compared to Oracle Version 4.1.1. Care should be taken in using this information because the products described will continue to develop over time. RTI updates this document periodically and appreciates any corrections or additions that can be brought to our attention.

2. ORACLE OVERVIEW

The following is a brief description of the components of Oracle and the subsystems of INGRES to which they correspond. When competing against Oracle do not fall into the trap of comparing functionality of components which are not *meant* to perform the same functions. For example, do *not* compare QBF with IAF – QBF is not meant to be INGRES's application development tool – it is merely a component of the complete application development package.

SQL == INGRES/QUEL

SQL is Oracle's query language. It was developed at IBM at the same time that QUEL was developed at U. C. Berkeley. SQL is the proposed ANSI standard.

UFI (User Friendly Interface) == INGRES Terminal Monitor (QUEL in 3.0 and later) and RBF

UFI permits you to run SQL interactively and includes commands comparable to our terminal monitor commands, with additional on-line help, output formatting commands, and prompting and variable substitution capabilities. With their UFI formatting commands they can do many of the things we can do with RBF.

Fastform == INGRES QBF

Fastform is like single table QBF. It gives you a default form which you can use as a starting point for a form in an IAF application.

IAF – Interactive Application Facility == INGRES – some parts of QBF, VIFRED and ABF

IAF includes IAG (Generator) to design the application and IAP (Processor) to run the application. Applications (which are just a series of forms – no reports, graphs, programs) are designed through a question and answer session and run by using keypad keys which are defined to be different functions (i.e., select, update, etc). Programs can be included in the application by using "user exits." Modification to the application is done by editing the text file that IAG creates or, optionally, restarting the question and answer session again.

Report Writer == INGRES/Report Writer

Report Writer includes RPT, which derives information for the report through SQL commands, and RPF, which formats the information based on formatter commands. The generation of reports is controlled by a single file, the report control file, which contains report writer statements for RPT, formatter commands for RPG, and textual information to be included in the report. It is known to be very difficult to use, but many of the simpler reports can be done in UFI. It does have a conditional and the ability to do multiple detail sections (include

multiple SQL commands for one report).

HLI == INGRES/EQUEL

In Oracle there is a precompiler for Fortran. Host Language Interface is done through calls for COBOL and C and "others." It is difficult to use. Oracle cannot call forms from within a program. Oracle does have the ability to do nested queries and to have multiple queries open (cursors). Some Oracle users use DEC's FMS (forms manager system) in HLI applications. Oracle claims 100% SQL/DS and DB2 compatibility for HLI programs. This is *not* true.

ODL - Oracle Database Loader == INGRES copy command

ODL is the utility for loading raw data from operating system files into a database. The commands are in a control file and run as a batch process. There is no **copy into** facility, or the ability to copy data from a table into a file.

INGRES Components for Which There Is No ORACLE Equivalent

- Visual Programming in general
- GBF
- QBF - defaults for multiple tables
- ABF with OSL - turnkey application builder including reports, graphs, etc.
- EQUEL/Forms
- RTIngres (switcher)
- INGRES/NET
- INGRES/Cluster

Oracle's Current Market Positioning

Oracle has positioned itself as the relational database management system that is compatible with IBM's DB2 and SQL/DS; all versions of Oracle on micros, minis and mainframes are identical. Therefore, Oracle enables you to standardize on software and connect your machines into a network that permit access to shared data elsewhere. Oracle advertises ability, compatibility, portability and connectability: a standard software environment for application development, network communications and decision support.

3. HOW TO COMPETE AGAINST ORACLE

Whenever possible when competing against Oracle, or any product, keep the comparisons at a high level – product history, product reliability, performance, technical superiority, ease of use, programmer productivity, and customer support and satisfaction. Oracle does not have a good reputation in these areas and will present a list of features and functions to the prospect which they have and INGRES does not to try to diffuse the major issues. Then we find ourselves on the defensive and waste time defending why we do not support such things as nulls, nested queries, horizontal scrolling, and record-level locking. Do not fall into the “trivia trap.” Respond by briefly addressing the missing features (see the “Abstract” and “Extended Response” entries in the following section) and present examples of the features INGRES has which Oracle does not (visual editing, multiple access methods, multi-table default forms, forms in its Host Language Interface).

The main point is that this game can go on forever, and on any detailed, feature-by-feature comparison there is no clear winner. Each product will always have things the other does not, and in the next release the lists will change. INGRES will have nested queries and nulls, Oracle will have graphics and 4GL. We can play this game if we have to, but it is time-consuming and only tends to confuse prospective buyers as they lose sight of the major issues.

Emphasize the broader major issues –

- Application development environment as an example of superiority in ease of use and productivity
- Performance and query optimization as an example of technical superiority
- Customer support and training
- Corporate philosophy
- Product enhancement history

4. FEATURES ORACLE STRESSES TO WHICH WE MUST RESPOND

4.1 Null Values

Definition:

A null value is a data value that is missing or unknown. It is different from all possible real values — not zero or blank. For example, in an employee record, a person's age may not be known; it is not zero. The need for nulls often arises when aggregates (sums, averages, etc.) are performed over columns with nulls values; it is desirable that the missing values be ignored in the calculation. When a table is created in Oracle, you may specify whether or not a column may contain nulls. If it can, then null is the default when a record is added with no data in that field.

Abstract:

Null values are controversial. Null values can be avoided by designing databases carefully. While meeting certain needs, null values can provide numerous opportunities for errors and confusion. Database authorities of no less stature than C. L. Date argue that a system of default values in situations where actual values are not know are preferable to nulls. However, there is no really strong argument against nulls, except that implementation in other systems has been far from satisfying and can lead to non-intuitive results. Nonetheless, this does not mean that RTI should not provide them. We certainly will be implementing nulls, but probably not until after Version 5.0, tentatively scheduled for May, 1986.

Extended Response:

The need for nulls can sometimes be avoided by normalization. Use rows where you may first be inclined to use columns and when a value does not exist there is simply no row, thus eliminating the need to deal with missing values.

In systems like Oracle and Database 2 a, three-valued logic is used to implement nulls — *true*, *false* and *unknown*. Because a three-valued logic does not always give you what you intuitively expect (and different people may have different intuitions), it is sometimes difficult to predict correct results. (This is a basic problem in implementing nulls for anyone.) For example, if a sum is done on a column containing nulls, we might all agree that nulls should be ignored, but what is the correct answer? Should it give a precise answer based on the non-null values when in fact the sum is not known if some of the values are not known?

Oracle has a null value function (NVL) which allows you to specify what value to substitute for null. However, it only works on numeric types, which is a major inconvenience. Oracle also has problems dealing with null values if they happen to occur in fields used in a calculation and the user has forgotten to use NVL. Users have reported that the query may go off and not return, or it may return a wrong value.

Oracle does not store nulls in the index. It would appear therefore that a query retrieving values that are null would have to do a sequential scan of the whole table.

For more information refer to Date's paper "A Critique of the SQL 'Null Value' Construct" in the appendix of this document and to the chapter on null values in the second volume of his book "An Introduction to Database Systems." There is an excellent example in the attached paper under section 5 on the problems of writing a correct query using nulls. I quote the abstract of the paper:

Null value support is commonly felt to be a desirable feature of database management systems. This paper argues that, on the contrary, such a feature – at least as it appears in today's SQL systems – is a mistake and the source of numerous difficulties for the user (not to mention the system).

Date concludes that default values are a better solution than nulls. A default value is a user-defined value that would be entered in a field whenever a record is added without a value assigned to that particular field. VIFRED gives you the capability of entering default values in fields. Again quoting Date:

The whole point of the default-value approach is that a default value is a normal value and behaves in predictable ways. It is therefore intuitively more satisfying than the null-value approach. It is true that it requires more explicit involvement on the part of the user; nevertheless, there is little reason to hope that further research on null values will lead to any significant breakthroughs. It is this writer's firm opinion that default values represent a vastly preferable alternative.

4.2 Nested Queries

Definition:

Make sure you understand what the prospect is referring to when he talks about nested queries – multiple simultaneously running queries (cursors) or an alternative way to flat queries of specifying joins.

Multiple open cursors are indeed very useful in many applications. They are needed when you want to retrieve and then do another retrieve or an update on the basis of the first retrieve. The most common example of the nested retrieve would be the master/detail query. You want to scan through a list of masters and at some point run another query to retrieve more detailed information. However, the query retrieving the master information is still open so that you can return to it to continue stepping through the masters.

Nested queries may also refer to a way of structuring SQL select statements. For example, the query "For each staff person retrieve all the tasks in which they are involved" could be written as a nested select from TASK within a select from EMP. See the example in the extended response below.

Abstract:

In response to writing joins as nested queries – we can do the same queries in QUEL without nesting. In both Oracle and RDB (which also has nested queries) there are performance penalties associated with using them. In response to the multiple open cursor question – we can automatically handle one level of nesting in QBF or ABF; more levels need to be handled in EQUQL. Multiple open cursors will be in Version 5.0, tentatively scheduled for May, 1986.

Extended Response:

Most queries that Oracle shows as nested queries can be written as "flat" queries in QUEL (and only sometimes as flat queries in SQL). (See Section 5.22, "QUEL," under INGRES's strengths.) This is demonstrated rather nicely by our QUEL to SQL interpreter, which rewrites SQL nested queries as flat queries.

For example, to find the person with the largest salary in each dept:

```

In SQL - SELECT ename, deptno, job
          FROM emp x
          WHERE sal =
                (SELECT max(sal)
                 FROM emp y
                 where y.dept = x.dept)

In QUEL - RETRIEVE (e.ename, e.deptno, e.job) WHERE
          e.salary = MAX(e.salary by e.deptno))

```

Now – which would you rather write? Another issue here is that it can be confusing to have two ways of writing queries in Oracle. *And* performance is dependent on the way the query is written in Oracle. Oracle recommends not writing nested queries, and in all benchmarks they reformat them as flat queries if they can. Having the capability to write nested queries is not wonderful if it is too slow to use. See also Section 8 on performance.

Nested queries in the sense of master/detail queries can currently be simulated in either QBF or ABF/OSL. In QBF you can do one level of nesting using all defaults – no code, no questions to answer; simply say you want to do a master detail query and name the tables involved. A default form is created with which you can retrieve, update, append or delete.

In OSL the query to retrieve orders and related parts for each order would be written as follows:

```
Menu item = {
  formname := retrieve(orders.orderno, customer.name)
             where name = customer.name and
                   orders.custno = customer.custno
  tablefield := retrieve(part.partname, part.cost)
               where part.partno = orderitems.partno
                 and orderitems.orderno = orderno
```

The second retrieve would be done based on each orderno that the first query retrieved.

QBF and OSL do not do real nested queries (they retrieve all the rows in the master table before retrieving the rows in the detail table, but to the user it looks the same).

More complicated examples can be done with EQUQL by first retrieving into an array or into a temp table and then doing the second retrieve or update based on the results in the array or the temp table. This is, perhaps, more cumbersome to code than if we had nesting but it *can be done*. In addition you lose the great feature of working on the one "real" copy of the data.

4.3 Portability

Definition:

Portability refers to the ease with which the system can be made to run on other computers.

Abstract:

INGRES is also very portable as you can see from the list of products on which we run. In the past, basic product development has been a higher priority than porting.

Extended Response:

INGRES is also very portable. It currently (April, 1985) runs on the VAX under VMS, Ultrix, and UNIX, and on UNIX on a wide range of hardware (Amdahl, Pyramid, CCI, Sun, Burroughs, NCR Tower, and 3Bx) In May, 1985, the IBM VM/CMS version will be in beta test. (Note that a nondisclosure agreement is needed before revealing IBM information.)

Future ports are planned. The difference is that we have in the past not depleted our VAX development resources by spending all our efforts on porting. It has been RTI's strategy to develop the fastest, most functional DBMS on the VAX before moving to other environments. Now that this is accomplished, we are spending more resources on porting, while in fact even more people than in the past are still working on basic development of better performance and new and enhanced front ends. (See Section 5.3 on product development history.)

In addition, it may take us longer to do a port because we will not take short cuts that may compromise the product when moving to a new operating system. For example, for the IBM product running under VM, we are not taking the easier implementation route of using shared global segments as Oracle did because using them has serious security problems. (See Section 5.1 on reliability.)

4.4 PC Product

Definition:

Oracle sells the idea that the same Oracle runs on mainframes, minis, and micros. Your same applications can be run everywhere.

Abstract:

RTI will offer a PC product in the three phases outlined below. These alternate solutions will allow the customer to select the option most appropriate for his application. RTI will not put the same code on the PC as runs on the mainframe (as Oracle advertises) because you do not need a mainframe DBMS on a micro, but we will offer end user compatibility across both systems.

Extended Response:

Oracle runs on the PC XT and PC AT, as well as Rainbows and PC-compatible micros. It is an "implementation of the mainframe Oracle DBMS." Its capabilities are restricted by the microcomputer's memory and disk storage capacities. Unlike PC/Focus is does *not* operate independently as a micro product. Rather, Oracle and databases are down-loaded through Oracle Link to the micro. Therefore, Oracle on a PC XT or PC AT can include an implementation of SQL and the set of integrated software tools for application generation, report writing, color graphics, and networking, limited by storage. As tools and data are needed, they are downloaded to a microcomputer through Oracle Link. There is no separate distribution media for the PC product. All Oracle tools are available on the PC, and reports can be composed interactively (UFI), and complex reports written with RPT. Applications can also be developed on the PC. You can start with a default screen or use a question-and-answer section (IAF).

Oracle claims that any program written for SQL/DS and DB2 can "run unmodified" on the PC. (We know this is not true even for the IBM mainframe!) And only "C" is supported on the PC.

Oracle Link allows data to be transferred between two Oracle DBMS's. You can download or upload data. What you can move are tables or views. The problem here seems to be that you can move whole files back – not just updated portions, so that you might run into concurrency problems. Other people could also have copies they are updating. Are whole files locked while the user has a copy on his PC? There is no way to run your applications on the PC accessing data on the mainframe so that the database system is controlling locking, etc. In other words, you need a duplicate copy of the data to use applications on the PC.

You can use the PC as stand-alone for data entry and transfer the data to the mainframe at 9600 baud. The price for Oracle Link starts at \$1000 for PC's and Rainbow type micros, and goes to \$6000 for the MicroVAX. (The MicroVAX is not on their list of computers on which full Oracle is offered.)

The first two implementations of INGRES on the PC will not have these problems. With our PC connect product, you will be able to download files in any format to use in conjunction with programs already running on the PC. For example, you will be able to download data to Lotus, and we will not force you to buy another product to use a spreadsheet with your data. Oracle is claiming to have their own spreadsheet package by June 1985. You will be able to upload files which can then copied into your database, but this will be under your program's control. The next phase of the INGRES implementation on a PC will be as a work station. This means that reports, applications, programs, etc., can run on the PC and access data on the mainframe. This may be a more satisfactory solution to some users than duplicating data on the PC and possibly running into data integrity concurrency control problems. Oracle does not currently have this option. Finally we will offer local data manipulation on the PC by allowing you to run INGRES stand-alone on the PC.

4.5 SQL – “IBM Industry standard”

Definition:

IBM (and Oracle) are trying to make SQL the industry standard. It is also the proposed ANSI standard for relational database definition and manipulation language. Oracle claims they are IBM SQL-compatible.

Abstract:

The reason SQL may become the standard is that SQL is backed by IBM. QUEL is the technically superior language according to database experts (Date, Codd). INGRES is moving towards forms interfaces, which alleviates the need for a query language. INGRES will soon provide the option of using SQL or QUEL.

Extended Response:

SQL is still only the *proposed* ANSI standard. There are a lot of QUEL users out there (university INGRES, commercial INGRES, Computer Associates, IDM). A lot of people (Date, George Schussel, who runs the National DBMS symposia, Dr. Codd) think QUEL is a cleaner and technically superior language. There are fewer inconsistencies and only one way to do a join (no nesting). The conclusion is that the reason SQL may become the standard is not because it is a better language than QUEL but because it is backed by IBM.

Oracle says SQL is IBM SQL-compatible – but there is no **union** operator in Oracle, and programming language interface, tree syntax, outerjoin, data types, and transaction specifications are all different. And these things are all guaranteed to be specified in any adopted standard. Oracle claims that programs written for SQL/DS or DB2 are 100% compatible and will run unchanged on Oracle. This is definitely *not* true. First of all, IBM publishes the incompatibilities between SQL/DS programs and DB2 programs, so they are not compatible with each other. Second, there are many examples of incompatibilities between the Oracle and IBM programs:

1. You must say “connect” in Oracle; not a command in IBM SQL.
2. Conflict between reserved words.
3. Host variables in DB2 do not need to be preceded by a colon
4. Error numbers for “no rows found” are different.

Many more examples can be supplied if needed.

Prospects will generally not buy based on query language alone. The total package – functionality, performance, reliability, visual programming and support should be stressed. With forms-based table creation and QBF, we begin to rely less on the query language and more on forms interfaces for the end user. In addition, there will be an optional SQL interface to INGRES. This is offered not because we believe it is a better language but because it is needed to sell an IBM product.

4.6 Clustering

Definition:

Clustering is a technique for increasing performance for very specific and predefined queries. Clustering allows rows from two or more tables that are frequently joined to be stored in the same physical space (page) where the values of the join columns are equal. For example, if you often retrieved the "dept" table along with the employees from the "emp" table who worked in a department named in "dept" (joined on "dept_no") you could store on a single block the row from the dept table with all the rows from the "emp" table where the "dept_no" in each was the same. Performance of queries that join these two tables will be improved (Oracle estimates as much as ten times) because it will reduce the number of i/o's needed to retrieve the rows. In addition, the key or join field ("dept_no") for all departments would be stored only once. It would not be repeated with each employee record. Clustering is transparent to the user.

To use clustering you must first create the cluster and indicate what the join columns will be – the cluster key. After the cluster is created, you can add any number of tables to the cluster as long as they have a column containing common data. The cluster key value is stored only once for all rows of any tables that share the same cluster key value. Clustering can also be used for a single table. This not only allows you to store data in sorted order (such as our ISAM or b-tree used as a primary storage structure would do), but it also reduces redundancy in storing the primary key – especially useful if the primary key (single or multiple columns) is long.

Abstract:

Clustering will help in limited static applications. INGRES's approach to optimization has been to build a sophisticated optimizer so that *all* queries will run fast. The bottom line is that INGRES wins benchmarks.

Extended Response:

Clustering is an optimization technique for limited specific applications. It is not a general purpose optimization strategy.

Clustering can cost you in performance for queries other than the query on which the cluster is based. In the above dept/emp example, if you wanted to retrieve just the rows from the dept table you would end up doing many i/os for each dept row. Departments will be spread over many more pages than if dept was stored as a separate table. Therefore, the simple query

```
SELECT *  
      from DEPT
```

would be slower than if you had not used clustering. You can think up many examples like this, such as changing the equijoin on the column in the cluster key to a greater-than or less-than join, and you can see that the cluster no longer buys you anything and will probably cause the query to run slower. Also, if all columns in the cluster key are not used in the *where* clause, you do not take advantage of the cluster structure. In conclusion, clustering works for only the one specific query it was designed to enhance.

In justifying the clustering technique, Oracle often uses the purchase order example. They show the number of i/o's required with and without clustering. *But* they assume you must go through an index each time to get every record, thus ignoring systems that have hash as an access method.

Oracle itself warns that you must carefully allocate space for clustering and choose cluster keys carefully. Too few rows stored with each value of cluster key wastes space. Too many rows with the same cluster key causes collision (all clustered rows cannot all fit on the same page), and overflow pages get created.

There have been many reports of problems with clustering (admitted by Oracle), but these are supposedly fixed in release 4.0. However, a recent benchmark using 4.1.1, where updates were done to clustered records, resulted in data corruption.

RTI has taken a more general (and more sophisticated) approach to performance – a state-of-the-art query optimizer. The optimizer works for all queries and takes the burden off the DBA for physical storage design and redesign if applications change. (See Section 5.5 on performance below.)

INGRES can cluster through unnormalizing data for static applications. Storage space required however would be greater than with Oracle.

4.7 Add a Column to a Table

Definition:

The ability to add a column to a table or alter an existing column without unloading the table.

Abstract:

The question to ask here is what are the consequences in Oracle after you have done the **alter table**? How do you get the data into that column, and what are the consequences after the columns are filled? (See Section 5.11 on space management.) INGRES allows you to change data dynamically; the difference is that it requires three commands and the overhead is paid up front rather than after the data changes.

Extended Response:

Oracle can add a column to a table with a single command, but it has to be the last (rightmost) column in the table. Oracle can alter an existing column, but is limited to expanding the size of an existing character column. It cannot in a single command delete a column, add one in the middle of a table, decrease the size of an existing column, or change the data type of an existing column.

When Oracle issues the **alter table** command, only the definition of the table is changed. You pay the price of physical rearrangement of the rows at the time you add data to the column you just added. For example, when you add data to the table, you just expand it, and the row no longer fits on the page on which it was stored. Consequently, it is moved to an overflow page. (Pages are 2000 bytes in Oracle.) Conceivably, you put data into these new columns, you will have lots of rows on overflow pages. Now what? There is no equivalent in Oracle to the INGRES **modify** command, and to rearrange the table (i.e., get rid of the overflow pages), you must unload and reload the table. So you might as well have done that in the first place.

In INGRES you can alter tables (add or delete columns, or change columns) in various ways. One way is to **retrieve into** a temporary table, making the desired changes, destroy the original table, and then **retrieve into** a table with the original table's name. This requires three QUEL commands. In Oracle, to drop a column or add it in the middle requires five SQL commands.

One reason you might want to add a column is to put data into that table that you had stored elsewhere. However, the **update** command in Oracle cannot update a column in one table with data from another table. For example, Oracle cannot do the following:

```
replace emp(floor = dept.floor) where
emp.dname = dept.dname
```

4.8 Multi-User Performance

Definition:

The ability to support many people working in the same database without degradation of performance.

Abstract:

Benchmarks have shown that INGRES can beat Oracle in multi-user environments. INGRES uses both local and global buffers. INGRES will have more processes running after five users; Oracle will have more processes running with less than five users.

Extended Response:

Because of their five shared processes, shared re-entrant server architecture and capability of record-level locking, Oracle claims they therefore perform better in a multi-user environment. Oracle claims they have references to prove it. In a recent benchmark (unfortunately one whom we cannot use as a reference), however, INGRES performed significantly better when five users were concurrently running a mix of queries against the same tables. Our performance will degrade less in a multi-user environment than Oracle's will because we do more overlapped i/o and cpu use. Theirs will remain the same for single and multiple users.

Oracle claims that INGRES does not have global buffers. INGRES in fact has both global and local buffer management. The implication of global buffers is that once a page is in memory, it can be used by many users, which reduces i/o which would be required for a user to get the page from disk.

In running IAF applications, it has been reported that 4 - 5 users of IAF really tax the VAX. (Watch the screen at 300 baud to see all their extraneous cursor movement.) They also recommend 1500 working set size extents for users of IAF - so it does not take many users to use lots of megabytes of memory.

(See also Section 4.13 on record-level locking because it affects multi-user performance.)

4.9 Outerjoin

Definition:

An outerjoin is a join of two tables in a database in which the values in the joining fields between the tables do not always occur in each table. This can occur because an errant value that has been entered into one of the tables has no match in the other, or because no data for a given value has yet been entered into one of the tables. For example, if there were two tables – “tasks” and “staff” with a join column “staff_id” – and you wanted to retrieve *all* tasks and all people assigned to each task you would write:

```
retrieve (tasks.all, staff.name)
        where tasks.staff_id = staff.staff_id
```

This would retrieve “tasks” as long as at least one staff person was assigned to it (there was a row in each table which satisfied the join condition). So “tasks” with no persons assigned would be missing from the result table. This is the case where you need an outerjoin – you want to list *all* tasks whether or not they have associated staff.

Oracle has an outerjoin (“+”) operator. To get the desired results in the above query you would write

```
Select .....
      from tasks, staff
      where tasks.staff_id = staff.staff_id(+)
```

Abstract:

The point to be made is that we can do outerjoins as a single command or as multiple commands. Just because Oracle has an operator does not mean they will perform them any faster than we do, and it is often the case that there are more efficient ways to do them than in a single query.

Extended Response:

INGRES currently has an outerjoin macro, which uses aggregates; this allows you to do the outerjoin in a single command. The above query using the outerjoin macro in INGRES would be written as follows:

```
retrieve (....)
        where outerjoin(tasks.staff_id, staff.staff_id, 0)
```

It is sometimes more efficient to do the outerjoin in a series of commands using **appends** and **deletes**, rather than one command with aggregates. Both the Oracle and INGRES outerjoins are not known for their speed. However, aggregate speed in INGRES 3.0 has improved considerably, and this has significantly improved outerjoin performance.

We also have an Application Note on outerjoins, which describes in detail the alternate methods of implementing them. We will implement an outerjoin function in a future release.

4.10 Tree Walks or Bill-of-Material-Type Queries

Definition:

In some applications the ability to perform a bill-of-materials (BOM) expansion is extremely useful. This is also called the parts explosion or transitive closure problem. Data used in BOM applications can best be stored as parent-child relationships (e.g., lists of part numbers and direct subparts within that part). The purpose of the BOM expansion is to find all parts which are descendant parts, descendants of descendants, and so on (building a tree). The expansion should be able to figure out the level of parts and check for cycles (whether parts are descendants of themselves).

Oracle has the ability to retrieve rows in a tree-structured order to satisfy the BOM application. To do this they use a **connect by** clause in SQL. The following example would retrieve all managers and all employees that work for each manager starting with the top manager and finishing with the lowest level of employee:

```
Select ename, job, dept, mgr
      from emp
      connect by prior empno = mgr
      start with job = "pres"
```

In addition the query could calculate and print each level and print out an indented bill-of-materials.

Abstract:

In INGRES you do not have a function comparable to the **connect by** function in Oracle. The EQUQL program needed to do the same thing is about 2 pages long, but guaranteed to produce correct results.

Extended Response:

BOM applications need to be implemented in EQUQL. We have a sample program which is available and also an Application Note describing an alternate implementation. These are not queries that someone is likely to write on the fly. If there is a need for this type of application, it is probably something that will be done over and over, and spending a little extra time up front to write an EQUQL program to optimize a particular implementation that is reliable is not too much of a hardship. The sample EQUQL program we have is about two pages long. Oracle tree walks reportedly have a few bugs, so the trade-off is reliability vs. ease of implementation.

4.11 User Friendly Interface (UFI) commands

Definition:

In Oracle's terminal monitor (User Friendly Interface) you have the ability to do formatting that is similar to what you can do with RBF. You can put titles on output, headers on columns, set various parameters, such as page and line length and break columns, specify aggregates, and format dates and columns. You can also set prompts and do variable substitution. There is a **decode** function, which allows you to divide data stored in one column into many columns (a fairly frequent request) or to combine data stored in many columns into one column. The **lpad** and **rpad** (left and right pad) functions allow you to create horizontal histograms. There is on-line help (a table that you optionally load into the database). And there is the ability to call either the system editor, which spawns a process like ours or a line editor, which does not. The editor you use is something that you have predefined and not an option you have at the time you invoke it.

The following are some examples of UFI commands:

Put a two line title on the result of a select:

```
UFI> title 'ACME CORPORATION || Monthly Report'
```

Put a column heading on deptno column, break on it and print sum of salaries by dept.

```
UFI> column deptno heading DEPARTMENT
UFI> break on deptno skip 3
UFI> compute sum of sal on deptno
```

Print hire date in the format: day of week, month, day of month and year.

```
UFI> select to_char(hiredate, 'day month dd, yyyy')
```

Set the page size to 25 and have it pause and wait before displaying each screen full of data.

```
UFI> set pagesize 25
UFI> set pause on
```

(There is a show command which shows the values of the current set commands.)

Append (insert) records into a table and have UFI prompt you for each value.

Preceding the column name with an "&" will cause a prompt for a value when you run the command.

```
UFI> insert into dept values ('&deptno', '&dname')
```

```
Enter value for deptno:
Enter value for dname:
```

Abstract:

We can do all of the same things but with a combination of the INGRES features. The ability to format the output of a query from the terminal monitor is convenient in many cases. The thing to stress here, however, is that with RBF you do not need to learn the syntax of a language because it is visually oriented.

Extended Response:

We can do most of the formatting with RBF that you can do with UFI. There are a few things such as **decode** and padding, which we would have to handle as part of the query. The cross-column tabulation would have to be done by the Report-Writer. The prompting and variable substitution we can do with macros (in addition, our macros have a conditional.) RBF is easier because it is "visual."

4.12 Triggers

Definition:

A trigger is a command (**retrieve, append, replace, or delete**) that, when executed, would automatically set off (trigger) another series of commands. Triggers can be used, for example, in implementing integrity constraints. In Oracle triggers are available only from IAF. The application designer specifies a series of commands that are executed before or after a commit is done. For example, after all fields in a block are entered and the **insert** command is selected, another series of SQL commands could be triggered. The triggered commands may be used to prevent the **insert** operation from proceeding, depending on some condition like the existence of some duplicate data. SQL commands executed after successful insertion of new data can be used to update or insert data in other tables in the database (an audit trail could be written since Oracle does not have this feature).

Abstract:

The reality is that neither system implements triggers in the real sense (at the database level). Both systems can implement them in an application with no host language programming. INGRES has a much more flexible implementation.

Extended Response:

The same thing can easily be done in INGRES with OSL or in QUEL by activating on a field or menuitem. This is more flexible than just allowing triggers on a commit. When either is selected you can specify a series of QUEL commands to always be performed, or be optionally be performed based on a condition. Oracle has no **if...then...else** logic to execute triggers optionally based on values, previous actions, etc.

4.13 Record-Level Locking

Definition:

Record-level locking means that each record (row) that is accessed is locked rather than locking the page that contains that row or locking the whole table which contains the row.

Abstract:

Oracle does record level locking only on the **read for update** command. Any other update command causes a table level lock. INGRES provides much better concurrency with page-level locks, has automatic lock escalation and more user-settable lock options.

Extended Response:

First, there is no indication that record-level locking is superior to page level locking. On the contrary, if many record locks are set, as in secondary key retrievals, the system may slow down due to the overhead that the operating system requires to manage the lock table.

Second, Oracle *only* does record level locking on the **read for update** command. It *appears* that they lock the rows as you need them, but when they actually go out and commit the update, they still lock the whole table. Oracle says **read for update** allows acquiring locks on rows, and the actual update is postponed until the end of the transaction. It "shortens the time the whole table is locked, by doing all updates near the end of transaction processing ..." Thus you can end up with lots of contention and lots of deadlock (which is, in fact, the case – from reliable source for Version 4.1). IAF also uses **read for update** when you retrieve a record-at-a-time to the screen. Oracle has no automatic lock escalation.

In *all* other cases (**INSERTS** and **UPDATES**), Oracle does table-level locking, which has serious implications in a multi-user environment, because any other reading or writing operation has to wait until the updating operation commits its results to the database – a feature they do not stress. INGRES primarily does page-level locking (2000 bytes of data at a time) with the option of letting INGRES decide when to lock a table (when overhead of page locking is great) or letting the user decide using the **set lockmode** command. Thus the number of rows which we lock depends on how many rows fit on a page (100 20-byte rows or two 1000-byte rows). In fact you can control the number of rows on a page to some extent by setting the fillfactor on the modify command. With a low fillfactor we can approach record-level locking, with a trade-off of wasted space.

Oracle also has user-settable locking. By default Oracle does not hold locks while reading a table (equivalent to our **set lockmode on table where level = table, readlock = nolock**). For consistent reads while others are updating, Oracle reads from the before-image journal file (but this file can get very large because they write it page-by-page). You can optionally lock a table in share mode (our default), shared update (simultaneous update – **read for update**), or exclusive mode (equivalent to our **set lockmode on table where level = table, readlock = exclusive**). They can qualify each with a **nowait** option. They do not have all our other options. Refer to the **set** command in the *INGRES Reference Manual* for details.

One other feature of Oracle's locking system – A **create** or **destroy** statement (or any other utility command) inside a transaction will cause a *database* lock to be acquired. *And* Oracle users have *one* database. The entire database is single-threaded on such a transaction. You can *not* do anything in INGRES which would require your whole database to be locked except recovery or backup commands.

(See more in Section 5.15 on locking under concurrency control below.)

4.14 Interactive Application Facility (IAF)

Definition:

IAF is Oracle's utility which allows forms based applications to be generated without programming; the result is an application which allows full multi-screen data entry and retrieval. It allows things such as field by field edit and validation checks and table lookup.

To implement a forms-based application in Oracle, you use IAG which steps you through a question-and-answer session in which you design the forms to use in your application, the edit checks, and the queries which will be executed by your application. The result of this session is a file which gets executed by IAF. Forms in Oracle are made up of blocks, which correspond to tables. Blocks are made up of fields. Applications are described block by block, field by field. The order of execution of the application may be the same as the order of definition or a one level menu may be set up to allow you to call any screen in your application. You must then return to this menu before calling another screen.

Abstract:

All you can do with IAF is define an application which you execute in the order defined or by calling all frames from a main menu. The application cannot use other tools of the DBMS and is difficult to change. It is probably fair to say that they cannot do anything with IAF that we cannot do using OSL and no EQUQL, and that we can do more with ABF than they can with IAF. With QBF and ABF they should not get to first base with IAF. IAF cannot touch EQUQL/Forms in power and flexibility.

Extended Response:

This response is primarily a list of things you *cannot* do in IAF. After reading these, the advantages of VIFRED, QBF and ABF/OSL should be obvious. At the end of this section is a simple suggested application – simple for INGRES and impossible for Oracle without using tools other than Oracle. It is suggested that you be prepared to explain how this application would be done in INGRES (better yet show them). Then ask your prospect to have Oracle do the same thing! The ABF Project Management demo application is also an example of an application that could not be done with IAF.

There is no conditional execution (no **if then else** as in OSL), which means there is no available branching or control of flow between forms in the application. You go forward through the application in the order it was defined – there is no backing up. Or you call all screens for a main one-level menu. For example, you could not branch to a second form based on some data that was entered and then return to the first form after you exited the second.

To go back and change an existing application involves manually editing the file that was created – a nontrivial exercise. The series of questions you are given to answer in the original session are dependent on previous answers and changing one answer in your text file may invalidate a series of questions which follow. You also have to remember to purge your old files, or you get lots of copies of an application taking up space in your directory. There is no automatic compilation and purging of old copies as in INGRES. With all pieces of INGRES, the method of creating and changing the application is the same.

Once you have the privilege to run an application, you can also go in and change the parameters that may have been set to define your environment and the resources you can use (e.g., you could up your working set extents to 16 to make your application run faster *and* use up system resources.) You cannot give different people privilege to execute different parts of the application. (Compare this to setting up symbols that start execution of ABF applications at different frames.)

You cannot leave the application and return to where you left off.

You cannot run reports from your application – or graphs if they were available. You can call a program from within an application by using user exits, but only at the same points at which you can define triggers – at commit points and not at the field level. If you run up against a limit in IAF, you are *stuck*. Because you cannot call a form from within a program, you must rely on tools outside of Oracle. You could, for example, use FMS forms in a program or use DCL to call a report from within an application. This requires, however, that you not only learn another system but also hope that the other system does not make changes in future releases which will affect your Oracle applications; when VMS went from 2.5 to 3.0 Oracle applications using DCL had to be revised.

You can define limited menuitems; you can set up one level of main menus from which you could call any screen. You cannot set up hierarchies of menuitems. Execution is through the selection of menuitems or keypad keys. New terminals may be defined to Oracle if you have the proper authority. The keypad keys can be redefined, but only at the DBA level for system-wide use – you cannot define your own.

You cannot easily share forms among applications. For example, you could not have a master form that you could use in various applications as needed. You need to create a copy of the form for each application.

Definitions of forms, applications, etc., are not stored in the data dictionary. You must manage them outside of the database system.

One nice feature they have (and stress) is that you can type in the whole **where** clause when running a query from a form. We can automate some of the things that they need the where clause to do – comparison operators in a field and pattern matching, and with table fields in QBF we can do the **or**-ing as well as **and**-ing of parameters in the fields of a form. You can implement accepting a string as a where clause in EQUQL, but it is not an automatic option on a form. The triggers discussed in Section 4.12 can only be activated by a “commit.” An application creator cannot enforce a commit at any point in the application (i.e., when a user is moving between blocks). It is up to the user to remember to commit.

Some more minor annoyances in IAF: Fields cannot span multiple lines; scientific notation is not supported; IAF cannot check against a valid list without a database retrieval; there is no escape to system editor for editing long character fields; you cannot scroll the whole form; and, deleted records remain on the screen until commit time. Before any message can be put to the screen, a database access must have been made – even if it is against a dummy table as they suggest in the manual.

The IAF TEST

This is a suggested application to be implemented with both INGRES and Oracle.

When you enter the application you have two choices: update a master detail form or run a report, which prompts you for input parameters. (Optionally a third choice would be to read your mail.) This would be the topframe in an ABF application with two (optionally three menu items). IAF cannot call a report or leave IAF and return without using facilities other than Oracle.

A requirement for the master-detail update should be that you do pattern matching and other-than-equality matches on a field in the form (e.g., > 200) and be able to use any combination of fields as input to the query. The simplest way to implement the master-detail update form in INGRES would be a QBF default. VIFRED could be used to make it flashier and to add edit checks. To push them really hard, make the requirement such that when you fill in an incorrect value in a field (such as an incorrect department name), you will be prompted to see if that is a new dept and, if so, it will display a new form to allow you to add the information for the new department name. Upon completion return to the update (calling) frame. This latter requirement can be done with OSL. It cannot be done in IAF.

The report can be done with RBF (use your imagination).

4.15 Multiple Retrieves in Report Writer

Definition:

Multiple retrieves in a report are necessary if you want multiple detail sections. For example, you want to list all tasks (master), all projects associated with that task (first detail) and all people associated with the task (second detail).

Abstract:

Multiple queries are useful in report writers. Oracle's report writer is much more difficult to use than ours.

Extended Response:

Multiple detail sections in a report are very nice. There is a way to do some of them using a conditional in the INGRES report writer. Oracle's report writer is very difficult to use and many more lines of code are needed compared to doing the same report in INGRES. A sample Oracle report is attached.

It is rumored that they are phasing out their report writer and replacing it with more formatting commands and a conditional in UFI. They have also hinted at a revolutionary new "user intimate" report writer in Version 5 – in mid-1985.

4.16 Horizontal Scrolling

Definition:

In IAF you can specify a field on a form which can scroll horizontally. This is useful if the data you are retrieving is too wide to fit into the field on the form.

Abstract:

It is more desirable to display the whole field on the form by allowing a column to wrap, than to use horizontal scrolling. We will, however, provide horizontal scrolling of entire forms in a future release.

Extended Response:

A field on a form in IAF is limited to 80 characters, and a field cannot span multiple lines. Therefore, you need this feature to handle wide columns. You can wrap columns in INGRES, which is preferable, because you can see the whole field on the screen at once. Oracle stresses this feature because we do not have it, but hopefully no one will lose a sale because of this one.

4.17 Soundex

Definition:

The soundex function converts similar sounding names to the same string and allows you to do comparisons against them. For example, you could search for every one who had a name that sounds like "schmidt" by writing:

```
select *  
from emp  
where soundex(name) = soundex('schmidt')
```

Abstract:

INGRES has more generalized and sophisticated pattern matching.

Extended Response:

What can I say? INGRES has very sophisticated pattern matching. We have everything they have (wild card matching characters "*" and "?"), plus "[]".

4.18 Larger Record Length and Long Text Field

Definition:

The ability to store fields and records longer than 2000 bytes.

Abstract:

The long field has limited functionality in Oracle.

Extended Response:

In Oracle you can indeed define a character field up to 64K (called long). The catch is that once you have it, you cannot do much with it. You cannot search on it (i.e., pattern-match against it). You can update it but must retype the whole thing. It is also not much use in IAF, because you can only look at 80 characters at a time. It would be difficult to edit in IAF; you cannot invoke your favorite system editor to edit a long text field as you can in QBF (CTRL-V). The largest fully functional field in Oracle is 256 bytes, significantly less than the 2000 character field in INGRES.

Miscellaneous Features

Oracle has the following

Functions – avg, sum, min, max, count

Arithmetic functions – round, trunc, abs, sign, power, vsize, to_char, chr, mod, floor, ceil, nvl

String functions – length, substr, instr, upper, lower, to_num, ascii, soundex, lpad, rpad, nvl, decode

5. INGRES FEATURES WE SHOULD STRESS

5.1 Reliability/Security

INGRES has a good reputation as a reliable system. Use references. Oracle has had little in the way of quality control. We do not have problems with data corruption, lost indexes or disappearing processes.

Basic system architecture features which compromise reliability and security are their use of one file (one database) for storing all tables and their use of shared global segments on VM/CMS.

INGRES places each table in its own operating system file. In the case of contamination, the problem would be contained to a single table. Because all Oracle tables are in one file, the whole database may be contaminated.

VERY IMPORTANT INFORMATION FOLLOWS: Oracle's implementation under IBM VM/CMS should be carefully explained, because it has *severe* security problems. Basically, anyone who has access to VM can read *and* corrupt data in an Oracle database. Oracle has taken a shortcut in their implementation under VM and uses a feature called DCSS – discontinuous shared segments. These are shared segments of virtual memory, which are accessible to all users. Oracle puts all buffers and lock mechanisms in these segments. All data being read and written to Oracle tables go through these buffers. Therefore anyone with a little knowledge of VM could read the data in these segments (i.e., read what was in the buffers at any point in time) and, *worse*, write to these segments (write over data in the buffers), thus corrupting data. You need no special privilege for displaying or changing DCSS's. Even passwords at some points pass through these buffers which, unless they are encrypted, means that you have *no* system security.

RTI has taken the approach that this implementation would be totally unsatisfactory and has implemented its own file system for INGRES under VM – a much longer and more costly development process. However, this process results in a completely secure system. Our front-end/back-end architecture provides an additional security feature because you have only your data associated with your front-end process, and you could not take a memory dump of your virtual address space and see someone else's data. In the future we will take advantage of the DCSS feature to store code which can be shared among multiple users, but it will be read only – and this *cannot* result in any security or corruption problems.

Another aspect of reliability is that when we implement a feature, we do it well and after careful planning. Although Oracle may have some features that we do not, many times their implementation, when looked at in detail, is incomplete. Examples would be their implementation of **alter table**, corruption of data on updates when using the cluster, and record-level locking for reading only. These things are obviously not demonstratable in a demo, and the prospect will not see them until he starts using the system himself.

5.2 Support

We have an excellent reputation for support. See the User Survey. This needs little comment. Reference current customers. Talk about the three training centers, scheduled classes and video training tapes.

5.3 Product Development History

Although Oracle has been on the market two years longer, INGRES has a more impressive list of added features. Oracle has put its efforts into porting. INGRES has devoted itself into improving functionality, ease of use and performance. (Where have you heard that before?) Oracle has not delivered (although they have been announced) any new major features in the last year. Compare this to the list for INGRES 2.1 and INGRES 3.0, all delivered within the past year – and on time.

5.4 4th Generation Language (ABF/OSL)

The addition of OSL to ABF marks the first time that a fully relational DBMS is combined with a comprehensive 4th generation development tool. You can develop complete turnkey applications with ABF integrating *all* parts of INGRES (reports, graphs, EQUDEL programs). This is far superior to anything Oracle can do with IAF. See proposed test under IAF above.

The points to stress are that you can

- Design frames in any order. (With Oracle you design them in the order in which they will be executed.)
- Test partially completed applications, which is great for prototyping. (In Oracle, to go back and change an application is different and more difficult than the original design session.)
- Modify applications as easily as create them.
- Include reports, graphs and programs in the application. These objects can either exist already, or you can create them from within ABF. You do not sacrifice the work you have put into forms if you need to move to more sophisticated environments (QBF to ABF to EQUDEL). If you design a form with IAF and run into something which you could implement with IAF, you would have lost the time you had put into designing the form because there is no place else you can use it. In ABF, you can share objects among applications.
- Control things like clearing fields, cursor placement, bringing up submenus, building queries from variable number of fields with comparison operators and pattern matching.
- Pass parameters and queries to new frames.
- Compile forms for fast execution.
- Let ABF handle all compiling and linking of applications.

5.5 Performance/Optimizer

INGRES allows you to choose from four different storage structures – heap, ISAM (indexed sequential access method), hash, and b-tree. (Compression is available as an option for all structures.) This significantly enhances your ability to tune the data base for performance. Different storage structures are suited to different type of queries and applications. For example, if you frequently access a table by exact matches on project number, you would want to hash in that column because this would be the fastest way to find the project you wanted to retrieve, faster than using an access method which has to go through an index, such as b-trees or ISAM. The important thing to stress is that all queries and applications will run *no matter what the storage structures are and no matter when you change them*. Changing structures affects speed of processing only. Beware: Oracle salesmen have been known to imply that this is not the case and that you have to modify tables constantly do different types of queries. They have also erroneously said that, when you add something to an ISAM table, it is not retrievable until you modify the table because the indexes are static. Everything is always retrievable, but after you add many rows to an ISAM table (how many depends on initial fillfactor), both the **appends** and **retrieves** will tend to slow because overflow pages had to be created to hold the new rows.

Oracle provides one storage structure – b-trees. In fact, the indexes are b-trees pointing to data stored as a heap. When you create a table in Oracle, the default structure is a b-tree with the first column as the default key. Remember this when doing benchmarks against them. The primary storage structure can be modified by the **cluster** command. This causes the data to be clustered actually by the key (like our ISAM and b-tree) instead of being a heap (sequential file). In addition to storing the data clustered on a key, the cluster key is stored only once, saving space for long keys. Oracle's data and indexes are stored in separate tables. Our indexes and data are stored in the same file, which is more efficient.

Within programs you can save QEP's for faster execution of repeated queries. Oracle does not have this ability and must reoptimize each time.

INGRES uses native (operating system) storage for numerics and uses system arithmetic operation utilities. Oracle converts numerics to an internal format, base 100 for storage, and must convert them and use their own routines for arithmetic functions. The latter approach is inherently slower. If you buy a floating point processor for your machine it will not help performance if you are using Oracle.

The INGRES optimizer is state-of-the-art and statistics-based. It derives query execution plans (QEP's) based on statistics available in the data dictionary, such as number of rows in the tables, number of pages of storage, storage structure and available secondary indexes; the needs of your query; and the estimated size of the result table. In all cases it picks a plan that will use the least amount of disk i/o and cpu time. INGRES can make sophisticated use of secondary indexes. In some cases several secondary indexes on the same table are used simultaneously in the same query.

To aid in optimizing queries, INGRES gives you the ability to look at the QEP's. This will help you tune performance because you can see what queries make use of keys and indexes and when sorts have to be done. This is the only system of which I know that has this feature. Demonstrating the set QEP is very impressive when shown to the *right* (technical) people.

Oracle uses a rather simplistic approach of determining the order of processing parts of a query. It assigns values to different parts (predicates) of the *where* clause, depending on whether keys or indexes can be used and whether ranges are asked for. In case of ties in deciding on the order of processing the query, Oracle picks a processing plan that depends on the order of fields in the target list and order of columns in the *where* clause. Thus, the burden falls on the person writing the query to know, for example, the most restrictive part of the *where* clause and to place it last.

For example you should write:

```
where sex = "female" and job = "manager"
```

and *not*

```
where job = "manager" and sex = "female"
```

Oracle does not store the number of rows in a table in the data dictionary and thus the optimizer cannot be smart about the order in which to do joins, in order to keep the amount of data in any temporary tables as small as possible.

INGRES also has the **optimizedb** command. This allows you to collect additional statistics (other than those normally stored in the data dictionary) on the distribution of data in specific columns in tables. When available, these statistics are used by the query optimizer to determine an efficient query execution plan.

If you have *ors* in the restriction in Oracle, the order of them affects query performance. If the columns in the *or* clause is other than the order in the table, performance degrades. Oracle does not optimize on *or* clauses.

Joins in Oracle can be written in two ways – as flat QUEL-like joins or nested queries. Again, the same query written in two ways can perform differently. In INGRES performance is independent of the way the query is written.

Oracle offers clustering as a performance tuning tool. In fact when their president, Larry Ellison, talks about how to make relational systems run as fast as other models, his one answer is clustering. Clustering allows data from two tables that are frequently joined to be stored in the same physical space where the values of the join columns are equal. Basically what they are after is keeping the number of file accesses (and disk accesses) to a minimum; this is also what out optimizer does. Clustering is transparent to the user. Rows from two or more tables are stored together where the columns of the tables corresponding to the columns of the cluster are equal. The cluster key value is stored only once for all rows of any tables that share the same cluster key. A join of the two tables "will execute as much as 10 times more quickly than a non-clustered join (assuming the join is based on the same columns as the cluster key)."

However, clustering will make other queries run much slower, because records belonging to a single table are spread over many more pages than would be needed without clustering. Space for clustering may be an issue, depending on row size of tables you are clustering. If not all rows with any given cluster key value can be stored in a single block, additional blocks are chained from the first block. "Most efficient use of clustering requires that all data rows sharing a single cluster key value just fill a cluster block." Therefore: "Choose space allocation parameters carefully. Choose cluster key columns carefully! Too few rows stored with each value of cluster key wastes space. Too much collision on many cluster key values causes excessive chaining of disk blocks."

5.6 Multiple Storage Structures

See Section 5.5 on performance. If you only offer one structure, the b-tree is probably the way to go, but b-trees are not always the best choice.

5.7 DECNET/Cluster

INGRES runs over DECNET and will work in a cluster environment. We will eventually network to IBM's, UNIX, 68000's, micros, etc. Oracle has made no statement of direction that we know about in this area. Because of the structure of their system, networking and distributed environments appear to be difficult for them to do. When they belittle our use of two processes per person, we can respond that this allows us to move fully into the world of distributed processing.

5.8 Visual-Forms-Editor

INGRES has visual editing. "What you see is what you get" when designing a form. Once a form is defined, it is stored in the database and can be shared among applications and users. With Oracle, you design forms (only in IAF) by answering questions. You must specify coordinates for each field on the form and therefore must have blocked out the form design on graph paper. You cannot play the game – let's see what this looks like, what if we change this to this? You have umpteen times the flexibility and ease of use in INGRES.

5.9 Multi-Table Query-By-Forms

With QBF you can get default forms for multiple tables with single- or multi-record display. Oracle has default forms only for single tables. It is impressive to show that, immediately after you create tables, you can start appending, retrieving and updating against them, using a forms interface and knowing no query language. You can now do **ands** and some **ors** between fields on the screen, comparisons and pattern matching by filling in fields – you do not need to type out the **where** clause. Be sure to show the **or** feature, because Oracle has pushed this capability in IAF in the past.

You can also use the default form in an application (good for prototyping) or change it using VIFRED. Oracle cannot change default forms or use them in IAF.

5.10 Integrity Constraints

INGRES can define integrities at the database level (**define integrity** command). These constraints will hold no matter which interface to the database is being used. The constraint on integrities is that they can only be single-table (single-variable) constraints. In addition, we can define validation checks on forms; these can be multi-table constraints. Oracle only has the ability to define integrity checks through the forms system (IAF); no constraints can thus be applied through SQL.

5.11 Space Management

INGRES has dynamic allocation and reclamation of space. In Oracle you have to estimate in advance the size to which tables will grow, and there is little or no reclamation of space – for both deleted rows and tables. They should really be hit hard on this issue.

In INGRES, when you create a database or a table, you *never* have to estimate how big they will get. They will simply grow until you run out of disk space. When you delete a table, the space is immediately available for re-use. In fact, you have the option of doing a **modify to truncated** command, which frees up all space for other use or a **delete** command, which deletes rows without freeing the pages. The pages are re-used, however, if records are re-inserted into that table. Within a page, if the table is other than a heap, all rows are automatically rearranged when rows are deleted or modified, so that all free space is at the end of each page. As new rows are added, they will be placed on pages that are not full before new pages are allocated.

In Oracle no space is reclaimed or re-used. Therefore you get a lot of space fragmentation, and if you have very dynamic tables, you must constantly unload and reload (which means shutting down the database); this is their only way to reclaim space. Oracle stores everything as compressed. If an update to a row causes the row to fit no longer in its original space, it is moved to an overflow page. So, for example, if you start out with lots of empty columns (or if you add a column with the **alter table** command), and then you put values in those columns with the **update** command, those rows no longer fit in the place where they were originally stored. Even if there is room on the same page, the row is moved to an overflow page. Oracle actually recommends that when you load a table, you might consider putting in dummy data in empty columns so that the space gets allocated for future replaces (!). In essence, this means you should not use nulls as your default. Oracle 4.0 is supposed to better handle space reclamation better.

When you create a table in Oracle, by default it goes into the "system" database partition. If you want your table to go somewhere else, you always have to use an appropriate space definition when creating a table. (See Section 5.16 on load/unload.) INGRES takes a better approach by allowing the default physical location at the database level while still allowing the default to be overridden at the table level. With Oracle, you must also set extent size – the size to which a table can grow. If these are exceeded, the table must be unloaded, new sizes specified, and the table reloaded.

5.12 EQUOL/Forms

There is no comparable facility in Oracle. On VMS you could use FMS with a host language program. On UNIX, what is there? But with INGRES you only need to use INGRES – everything is self-contained. And you do not need to worry if your applications will continue to run when changes are made to those other systems you may have had to use.

5.13 Preprocessors

The full facility and syntax of QUEL is available to high level programs. Once you learn QUEL, you can use it from the terminal monitor and within all five programming languages (Fortran, C, COBOL, Pascal, and Basic) without learning an additional call interface. You can run the queries from the terminal monitor for testing before putting them in your program. It is estimated that the same programs in Oracle require at least twice as many lines of code, in addition to learning the call interface.

Oracle does offer a preprocessor for Fortran, but as recently as February, 1985, a user told me it was too buggy to use, and they were still using the call interface. Using the call interface requires the programmer to worry about lots of details that are otherwise transparent, such as data type conversions between database columns and program variables. There are some extremely difficult issues of which the programmer must keep track when making changes to the program, such as position-dependent binding. Therefore, with INGRES you get faster application development and more productive programmers. Also, the run-time libraries can change to add functionality and performance without requiring that the user change any code because the call interface is hidden by the preprocessor.

5.14 Graphics

INGRES has it; Oracle does not – yet. Oracle announced graphics well over two years ago. They were basing their graphics package on a Precision Visuals package, but have since changed twice. It will do more sophisticated graphs than our current package, but is not supposed to be well integrated. It was announced as part of 4.0, but did not make it. It will be priced separately. We too will offer a more sophisticated, higher-priced graphics interface tentatively within 1985.

5.15 Locking/Concurrency Control – Page-Level and Settable Read Locks

INGRES does page-level (2K pages) locking and has user-settable locking. Except in *one* case (**read for update**), Oracle does table-level locking; **read for update** causes record-level locking. Oracle also has user-settable locking but with fewer options. See Section 4.13 on record-level locking above.

RTI believes that page-level locking is superior to record-level locking for the following reasons:

- At the operating system level, i/o is page oriented. Therefore, there is less overhead with page level locks.
- It is often desirable to reorganize records on the page during an update to avoid fragmentation and maintain high disk space utilization. If the lock is maintained at the page level, there is no additional overhead to obtain locks for those records on the page being reorganized. Oracle does no reorganization, and the result is lots of wasted space.
- INGRES maintains b-trees and ISAM indexes to the page level. This allows the index to be smaller, which increases search performance. Also, when records on the page are reorganized, there is no additional secondary index maintenance required.
- INGRES also provides a hash access method. Hash indexing always points to a bucket (a page). Maintaining the locks at the page level is required because the page must be searched to identify the proper record.

Locking is purely a performance issue. When evaluating a system, one must look at overall performance. If a system that does record-level locking is slower than one doing page-level locking, then locks will be held longer, and you may not buy any increased concurrency. The longer you hold the locks the higher the chance of collision.

INGRES has automatic deadlock detection; Oracle does not have true deadlock detection but uses a timeout mechanism. Therefore, if the system is slow and Oracle sees nothing has happened, Oracle may detect it as deadlock and abort your transaction. INGRES has the option of setting a timeout when waiting for locks. INGRES also has the option of setting **no deadlock** which guarantees there will be no deadlock at the expense of decreased concurrency, because with the **no deadlock** option there is almost exclusive table-level locking.

5.16 Load/Unload Utility

The INGRES `copy` command is easier to use, more functional and faster. Oracle can only load from fixed length records. (They may have added a variable length loader.) INGRES can copy from variable or fixed length fields and records with various delimiters between fields and records. INGRES can also copy from binary files. Several benchmarks have shown that the INGRES `copy` command is two to three times faster than Oracle. The INGRES `copy` command operates in one step and is executed from within INGRES. The Oracle copy is done using the ODL (Oracle data loader) facility outside of Oracle. You put the required three commands per table to be loaded into a file and run it as a batch process. You must first specify what the input file looks like, define the mapping to the database table, and then specify the command to load the tables.

Compare in Oracle:

```
Define record employee as empno (char(9)), empname (char(10));
Define source file1
  length 19
  from raw data file
  containing employee;
For each record
  insert into emp (empno, empname):
    <empno, empname>
Next record
```

Compare in INGRES:

```
copy emp (empno = c9, empname = c10) from "dra0:[demo]file1"
```

Oracle does have the ability to load values into fields as part of the load. They also keep a log file of rejected records. After a specified number of rejected records are found, the load will abort.

INGRES can run the `copy` command in the other direction. You can create a binary or ASCII file (fixed or variable length) from the data in a table. You cannot copy out to an ASCII file with Oracle. (See also Section 5.18 on recovery.) If you need a readable file, you can get around it by spooling SQL results to a file, but even if you take off all headings, it puts extraneous lines into the file, which makes it difficult to read back into a table. (I know – I tried reading output into INGRES.)

5.17 Integrated Data Dictionary

In INGRES everything – from tables, to permits, to integrities, to forms, graphs and applications – is stored in the dictionary. For example, when you define a form, the specifications are stored in the database. As you make changes, the data dictionary is updated, so you automatically have the latest form, table, etc., available. You can always find out what objects are available, who owns them, and other information. Because the data dictionary is made up of tables, you can query them through QUEL or QBF.

Oracle stores only information about tables in the database. You have to manage reports and applications yourself. Oracle allows you to query the data dictionary only through views. You do have the ability to add fields to these views, for example, if you want to add a descriptive column describing each column in your database. You could do the same thing through a view in INGRES.

5.18 Recovery/Audit Trails

INGRES has various utilities for backup and recovery. The **restoredb** command recovers from a soft crash or any abnormal abort when you are within a transaction in INGRES. The **recoverdb** command resurrects the database from a hard crash. The **recoverdb** command can recover from a checkpoint or recover using the journal files. The **ckpdb** ("checkpoint the database") command allows you to take a static picture of the database as it exists at any point in time. If you need up-to-the-minute recovery on a table, you enable journaling for that particular table. This creates a journal file of all rows that are changed. In addition, the **copydb** and **unloaddb** commands make copies of a database (including permits, integrities, indexes, etc.) or parts of a database. The **copy** command allows you to write out an ASCII (readable) file or a binary file. Because INGRES tables are stored as VMS files, you can also use the VMS **backup** command.

Oracle has an **export** and an **import** command for backup and recovery. The **export** command can be run only by the owner of the tables and views. Therefore, only one user's data can be unloaded at a time. **export** produces a non-readable, non-modifiable read, and it does not record permits, indexes or cluster specifications. The **import** command cannot selectively import tables and can only import a single user's data at a time. You cannot export directly to tape; in INGRES you can automatically write a database checkpoint to tape.

The log file that Oracle writes in order to roll back a transaction is a "before image" file. Oracle writes out the page that is changed – not the record as INGRES does. Oracle also has an audit trail capability. It is *very* slow. You may suggest that prospects try it if they need up-to-the-minute recovery. Oracle suggests you can keep your own audit trail by using triggers in IAF to write changes to a table you create and maintain.

5.19 Default Forms, Reports, Graphs

With INGRES you can get default forms, graphs and reports. Oracle does have a default form (Fastform) capability for a single table. However, this form cannot be changed or called up from within a program.

5.20 Transaction Management/Savepoints

At the end of each transaction in INGRES (whether single- or multi-statement), the user is guaranteed that the changes have been written out to disk. If there is a crash, you can recover your most recent transaction. In Oracle this is not the case. They do "write behinds," which means that they do buffer-writes and do not guarantee that, when you think the transaction is done, that it is really written to disk. You must explicitly commit a transaction or turn "autocommit" on. This is important to remember in benchmarking. To compare like with like, we should benchmark against Oracle with autocommit on. Because they do not write out changes unless you specify that they do so, we do not think they do transactions "right."

Within a multi-statement transaction in INGRES, you can define savepoints. These are markers to which a partial transaction abort may be executed later in the transaction. At any point in the transaction, a user can abort to a pre-defined savepoint. This is very useful in a large transaction with lots of options. Suppose you have worked your way down one possible route in the application and find out that you cannot continue. Liken this to making airline reservations for various legs of a trip. You do not want to undo all the reservations you have just made when you find yourself in Cincinnati, Ohio, and cannot get out. You want to back out partially (to Chicago) and choose another route from that point on. Without savepoints (Oracle does not have them) you must rollback all or nothing if a certain condition within the transaction fails. This could result in lots of redundant coding and redundant operations against the database.

5.21 Report-By-Forms/Report-Writer

INGRES has default reports and RBF – again, “what-you-see-is-what-you-get.” Oracle has neither concept. They do have fairly sophisticated formatting options in UFI. (See Section 4.11 on UFI above.) The capabilities of the report writers in both appear to be about the same, but the INGRES Report-Writer is much easier to use, and many fewer lines of code are needed. Oracle does have the nested query ability to make doing reports with multiple detail sections easier. There are ways of doing this in INGRES, but it requires building a temporary table first. Sample reports are attached.

5.22 QUEL

QUEL is cleaner than SQL. There is one way to write joins (really an implementation issue rather than a language issue); query processing is independent of the order of restrictions; you can replace from one table into another; and, there is nesting of aggregates, **retrieve into**, and more sophisticated pattern matching than SQL. The **retrieve into** should be stressed as a way to limit data that future queries will have search. It is often a preferable solution to nesting queries. One way in which **retrieve into** is cleaner than SQL is that **where** clauses can be applied to rows being retrieved or rows to which an aggregate will be applied. In SQL the **where** clause restricts rows retrieved, and the **having** clause restricts aggregates.

5.23 INGRES/MENU

The new 3.0 subsystem interface, INGRES/MENU, is a visually oriented, forms-driven interface to and between all INGRES capabilities. It allows you to start up the terminal monitor and any one of the INGRES "front ends" without starting a new back end process. This saves time and resources and improves performance. It is a very nice tool for the end user. In addition, it maintains a history of your session and a forms interface for table creation. Oracle has nothing like it. Stress the easy table creation.

5.24 Macro Utility

INGRES has a macro facility, which can be used from the terminal monitor. You can set up canned queries or redefine the query language. You can have it prompt you for input and execute based on a condition. For example if the user wanted to type "get" instead of retrieve:

```
{define;get;retrieve}
```

would be the macro.

Oracle does not have a macro facility. Some of the ability to set up prompts and canned queries can be done through UFI commands, but you could not redefine the language or use conditionals for execution.

5.25 Price

We hope that all of you all know the current price of INGRES. The current price of Oracle on a 780 or 750 is \$48,000. They also have multiple-copy pricing and special prices for OEM's. Supposedly they have "cut" their prices in the past by not listing an interface such as the report writer on the invoice.