# Oracle Configuration for Intelligent Networks Applications

Oracle Configuration for Intelligent Networks Applications

August 1998

Authors : Yogish Pai
Joe Cittern
Steven Hendrix

# Contents

# 1. Introduction

This document describes how Oracle Server can be configured to meet near real-time requirements for IN applications. The document is based on Oracle's recent experience with IN vendors, telecommunications companies and end-users. Many of these telecommunications companies and IN vendors are now discovering the advantages of using Oracle's open technology in the previously proprietary world of Intelligent Networking databases and service delivery, and in the open world of the Internet.

The telecommunications industry is experiencing fundamental change, caused by many factors, in particular;

- Continuing deregulation, leading to increasing competition from new technology start-ups
- The requirements for new services, in order to increase usage and retain customers
- The dramatic rise in mobile telecommunications over the past few years.

In conjunction with these changes, a revolution is occurring in the previously closed IN environment. The dynamic, open world of the Internet and its related technologies are beginning to make a significant impact upon the telecommunications services that are being delivered to customers, and the technologies that are being used to deliver them.

There is now no doubt that Internet technology and services will have a dramatic effect upon the way that telecommunications services are implemented, delivered, provisioned and managed. Oracle is empowering its customers to be at the forefront of this revolution.

Oracle is now providing its customers a unique capability to leapfrog traditional IN service delivery models. Oracle products are being used to deliver the 'traditional' IN-based services, and also, increasingly, the newer, Web-based or focused services. In many cases these new services will supplement or replace the older services, using an open, reliable, scaleable, and flexible infrastructure that will support the increasing complex and competitive telecommunications environment . This environment must meet today's requirements to support highly performant and reliable services, and tomorrow's requirements to be Internet capable, and fully integratable with new technology such as JAVA and Object technology. These capabilities were not even thought of when IN standards were first developed, but today's telecommunications environment must support them.

By providing all these capabilities to its customers, Oracle is enabling them to deliver new, advanced telecommunications services to market quicker, cheaper and more reliably than ever possible before.

## 1.1 Traditional IN Service Delivery

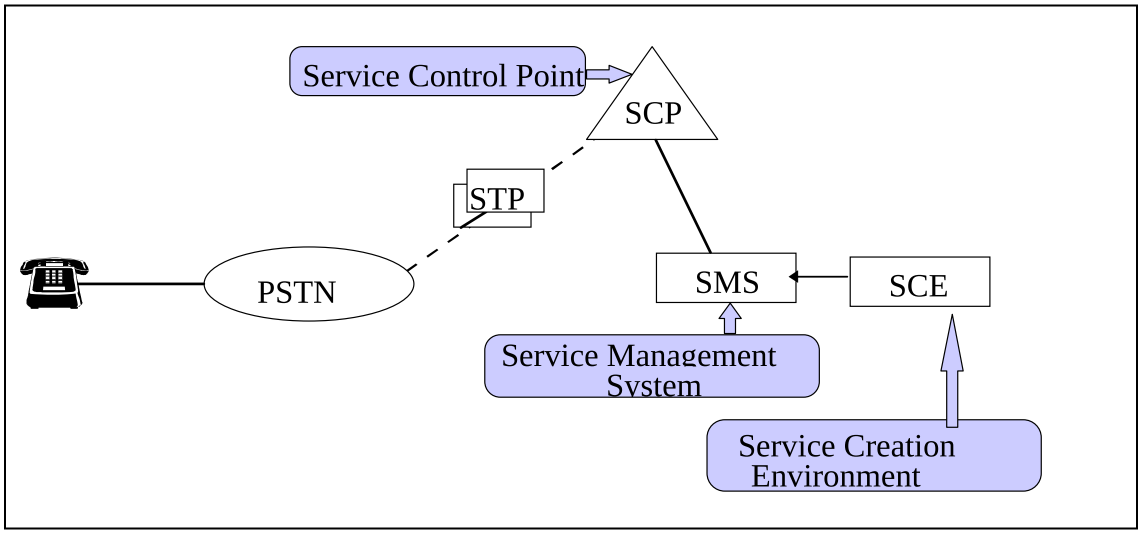

**Figure 1: Traditional Intelligent Network Architecture**

The above diagram illustrates in outline the traditional Intelligent Network architecture. Intelligent Networking is an industry defined mechanism for extending the capability to manage telephone calls, by migrating much of the 'intelligence' used to route calls out of the switches and into programmable platforms. IN defines the standards used for communication between logical and physical entities used to process telephony interactions, including calls and associated services. It was created to make the system independent of particular service providers or Network equipment vendors, and allow faster delivery of services based on open standards.

In theory therefore the various IN architectures (there are a number of variants, in time and geography) allow for dynamic creation and modification of services based around a set of standard protocols and message sets. Services are the revenue stream of business for the telecommunication industry, where time to market and bundled services are key to generate new customers and increase business with existing customers, whilst keeping existing customers. Within these self-contained environments, new services are defined and created within the Service Control Environment (SCE), and then downloaded to programmable Service Control Points (SCP), via the Service Management System (SMS). The SMS is also used to provision the data for services.

In practice, the reality of IN has not fully met the promise of its originators, for several reasons, including;

1. Much effort was focused on Service Creation aimed at the SCP, omitting the provisioning aspects of service delivery.
2. Standards were slow to be agreed and slower to be delivered, and incompatible variants have arisen, increasing complexity.
3. Many of the services are too difficult to use. DTMF is not the ideal user interface, but is generally the only one defined by standards. It is not uncommon for a service to require the user to enter 20 or 30 digits to control one aspect of it.

Combined with these factors is the fact that most IN systems use specialised databases and programming languages or constructs, increasing incompatibilities between different vendors equipment, and also meaning that skills and applications cannot be easily transferred between operators or vendors.

Oracle's technology is today being used as a more cost effective and faster way to deliver services for these traditional architectures, and also to enable these and newer services to co-exist and interact with Internet based applications. It is Oracle's belief that the new architecture now emerging, based in large part upon standards derived from Internet technology, will lead to many of the above problems being overcome.

## 1.2 The New Model for Service Delivery

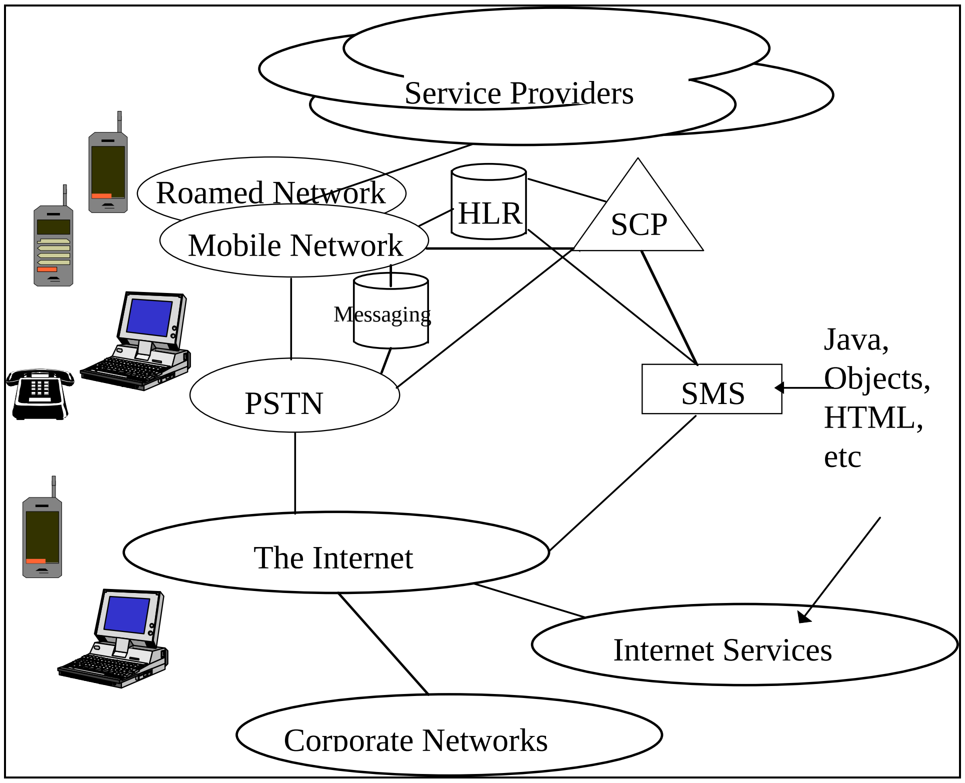

**Figure 2: The New de-Facto Intelligent Network Architecture**

Figure 2 shows the new, de-facto architecture developing in response to the proliferation of mobility, combined with new Open Internet technologies and advanced handset capabilities. Much more complex and dynamic than the traditional model, it is a model where the boundaries of Mobile and Fixed, Intelligent and Switch, Computer and Handset, are blurring and will eventually disappear.

The ubiquity of the HTML standard, handsets with in-built web browsers, services designed for self provisioning via the Internet, and the arrival of programmable, downloadable intelligence within handsets and other network devices, are some of the key technologies that are transforming the way telecommunications services are delivered to customers.

These technologies allow customers to bypass many of the services that traditionally could only be provided by the network provider, and, crucially, they also bypass the traditional IN function of determining when and how service selection is made. For example, when the handset can be programmed with a number of different service addresses, and make the choice for the user, or when a mobile handset performs voice recognition and dials the required person or service, the functionality that was previously envisaged as being centralised is no longer needed there.

As this intelligence moves out of the Intelligent Network, into terminal devices and Internet/Intranets, it will have a far-reaching implication for service-delivery and capability.  This is shown in Figure 3.

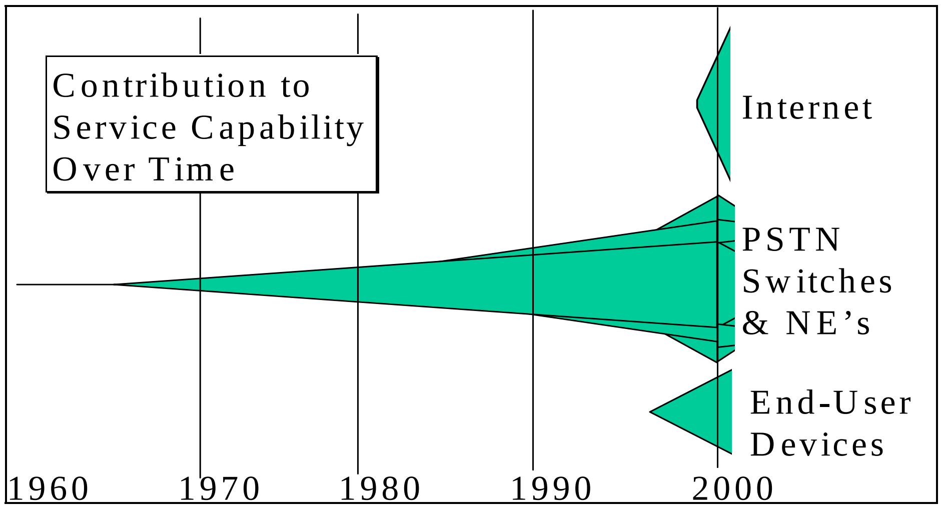

**Figure 3: Movement of Intelligence out of the Network**

Note also a key difference between the traditional IN architecture and that shown in Figures 1 and 2.  In the traditional model, the entire environment is within the realm of a single telecommunications company, which is therefore able to control most or all of the aspects of service delivery.  This is no longer the case in the new model, and therefore much more of an emphasis is placed upon the capability of the operators to provide and support standards-based interfaces and capabilities, in order to allow easy interoperation with other service providers.  Oracle is currently providing this capability within the Computing environment, and is now starting to provide the same commonality within the Telecommunications environment.

## 1.3  Synergy With Future IN Direction

IN standards are moving forward, in Europe with CS2, CS2+ being the closest to release, along with CAMEL to provide a formal mechanism for IN within GSM.  Without doubt some vendors and operators will release services based upon these standards, but at the same time, an increasing number of operators, frustrated by the incompatibilities and slowness to market of IN standards are taking a more pragmatic, Service Node[1] or Internet-based approach.

Looking further forward, the vision put forward here by the authors is very much in line with the direction taken by TINA-C. This consortium put forward in the middle 1990's a future vision based on what was then very much a state-of-the-art computing direction.  It encompassed distributed processing, objects and had a more comprehensive  focus on management than the original IN.  The new direction that is now being taken by the defacto adoption of Internet based standards such as CORBA is very much in line with TINA and their view of where IN is heading.

[1] Whilst Service Node is often referred to as 'IN' by those selling or implementing it, to a large extent, Service Nodes bypass much of the CCF and SCF and thus move the intelligence out to the periphery of the network.

## 1.4 Conclusion

In tomorrow's Internet-focused world, the user will take much of the responsibility for determining which services they use, even if this is made transparent to them, and the Telecommunications Companies and Service Providers that succeed will be the ones who live within and enable this new mode of service delivery. They will succeed by providing customers with valuable, easy-to-use, self-provisioning services that therefore increase the amount of bandwidth, time and functionality that customers wish to spend.

The rest of this document is in three sections. Section 2 describes how the telecommunications companies can benefit by using Oracle technology in the previously closed 'traditional' IN model. Section 3 describes how customers can extend these benefits to the new service model where interaction with Internet technology and services is key to the service. The following sections provide more detail on Oracle technology and specific applications of this technology to a high throughout, rapid response time benchmark conducted with various IN platform vendors.

# 2. Technical Details

The IN data management layer handle two types of transactions that can conflict with each other. These are call processing, and data provisioning transactions. The objective of the management layer is to manage transactions in such a manner that they meet the system requirements while supporting the response time and throughput requirements of both transaction types. At the center of these two transactions is the SCP containing the database which contains all the required information for handling the calls, plus the associated applications which actually contain the logic necessary to implement the service, or that part of it which resides on the SCP.

## 2.1 SCP (Service Control Point)

The SCP contains the service logic, and stores the call control and call routing instructions for the delivery of the IN services. The SCP gives the service provider the theoretical capability to rapidly adapt services to the customers needs and to meet market demands, something that can only happen in practice if it contains a flexible and adaptable data model.

As the SCP database contains all the routing and delivery information, it is important that the SCP is able to meet performance and availability requirements. Early IN implementations typically were able to cope with the unavailability of the SCP, but today this is usually not the case, and if the SCP is unavailable the service usually will be as well, whether that service is an advanced feature, or a basic fundamental such as call set up or mobility support. Because the reliability requirements for different operators and services are different, Oracle provides a range of solutions to address reliability requirements. These are wider than just fault tolerant nodes, since the common use of a fault tolerant system to address these requirements does not, at all address the need for disaster recovery or geographic independence.

These cover protection from four types of failure:

- transaction
- media,
- node
- disaster/highest availability

and in conjunction with availability also provide a number of solutions to scaleability issues.

### 2.1.1 Transaction Recovery

the function of transaction recovery is to ensure that the database remains logically consistent in the event of a failure, or other event, that causes some data updates that are part of a logical set not to be completed. The Oracle Database's transaction protection facilities will ensure that either all data updates done as part of a logical set are completed, or none are, and any incomplete updates are backed out.

This will be increasingly important in the future, as users access and change more data, and Oracle's unique Web-aware transaction protection ensures that updates completed as part of a series of web originated updates will be logically consistent.

As would be expected from the world's leading database, the transaction recovery mechanisms are fully capable of supporting the high update rates that might be experienced in a high volume environment.

### 2.1.2 Media recovery

The Oracle database manager is aware of, and cooperates with, a wide variety of hardware systems from all the leading vendors. It encompasses dual and triple mirrored disc, RAID, multiple ported controllers, remote and local disc arrays, and a variety of backup devices. In all but the most extreme circumstances of multiple failure, media failure will be invisible to applications and will be automatically recovered on line with no down time.

### 2.1.3 Node Availability

Oracle works with leading vendors such as Sun, HP and Sequent to provide high availability through a number of techniques, including SMP, clustering and Oracle Parallel Server (OPS). OPS works in conjunction with hardware clustering technology to ensure that in the event of a CPU or system failure, other nodes in the cluster continue to process and have access to shared data. Depending upon the circumstances and configuration, there may be no downtime at all, or just a few minutes.

### 2.1.4 Disaster/Highest Availability

For those applications needing the highest possible uptime, high availability or fault-tolerant systems are insufficient. The application needs to be able to survive in the event of a complete system failure. This might be caused by local factors, such as hardware or software failure, by operator error, or by catastrophic failure, such as fire, flood or electrical outage.

Oracle8 contains a range of capabilities aimed at providing resilience to cover all these failure scenarios. These include standby databases, replicated databases, support for multi-site mirrored discs, or combinations of all of these. Oracle can therefore be sure that it can configure an architecture to support just about any set of failure scenarios.

Equally importantly, Oracle has the skilled and experienced consultancy available to help its customers plan, design and implement such systems. To provide the highest levels of availability, more than technology is required.

One of the standard ways of providing high availability of a service is the mated-pair solution, this is extremely common with the telecommunications environment, and is recognised by some standards which allow mate nodes to ‘talk’ to each other, (even though these are today rarely implemented).

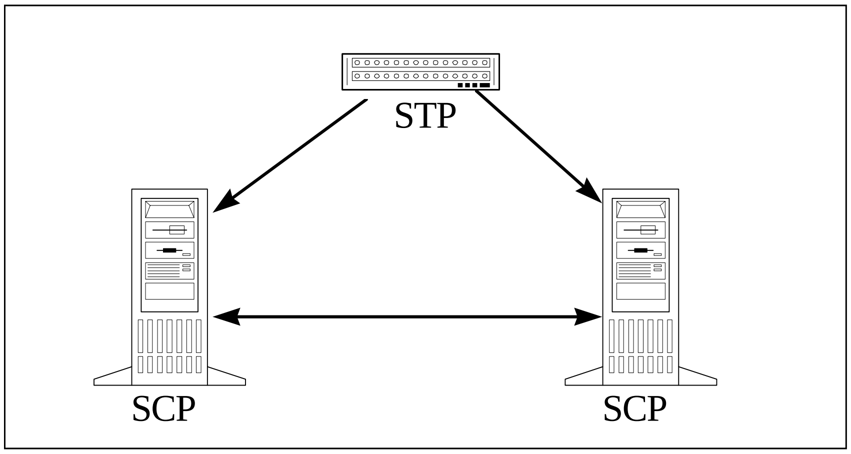

**Figure 4: Mated-Pair Approach**

Figure 4 represents the mated-pair approach implemented in a network. The STP sends the transaction to either of the SCPs (located at two sites connected over a network). If the STP does not receive a response within a pre-determined time, it registers the SCP as being down and transmits the request to the mated-pair. Alternately, the STP can be configured to send the transactions to either of the SCPs in a round robin fashion. For either of these solutions, the requirement of this approach is that both of the SCPs need to be synchronized within a specified amount of time (ranging from 1 to 10 seconds - depending on the application). This is achieved by replicating data between the mated-pair in real time.

Oracle provides three types of replication, each slightly different and suitable for different service scenarios. The customer can therefore choose the configuration best suited to their service without needing to compromise.

## 2.2 Call Processing (SCF)

The call from the SSP is forwarded via the SS7 network to the SCP embedded within a Transaction Capabilities Application Part (TCAP) message. TCAP is an industry standard of defining the messaging for call processing. The SS7 network should be able to handle data depending routing. This will enable the service provider in configuring the service dynamically, without having to shutdown the service. The industry standard for data distribution routing is handled with the System Signal 7 (SS7) protocol. The data dependent routing will be based on a predetermined parameter such as the actual service being provided (example 800 # service) or partitioned based on telephone numbers (example, east cost & west cost area codes, etc.). The basis of partitioning the service will be dependent on the number of calls that need to be processed per second. In addition, an Internet Protocol (IP) approach is now starting to appear for some network devices, especially as interaction with Internet services becomes an integral part of newer services.

## 2.3 Provisioning Transactions

The second type of transactions impacting the SCP are those used to provisioning or update the service. These provisioning transactions are used to update the call processing data, usually initiated by the creation, update or deleting of a subscriber or service in the system. This is typically initiated by a Subscriber Management System (SMS). Note however that in many scenarios, and increasing so, data may be updated by the user. In the classical IN this was known as 'Non Standard Update', perhaps hinting how often it was expected to occur! This is no longer the case, and the platform must also have the capability to manage network-generated updates. This may mean, for example, the requirement to reflect a change back to a management system, so that it can be held in a central 'master' system, or so that it can be replicated to other geographically separate nodes. The SCP should also be able to handle the provisioning transactions initiated by one or more SMSs and to manage the case where updates may conflict. This may require embedded logic within the database to manage consistency and manage conflicts, or understand who holds the control for any particular data entity. The general business rule, in a case where multiple SMSs send transactions to the SCP, requires that the SMS can modify only subscriber information that it owns, i.e. the SMS should have created the service/subscriber, but the system needs the flexibility to manage other models as well.

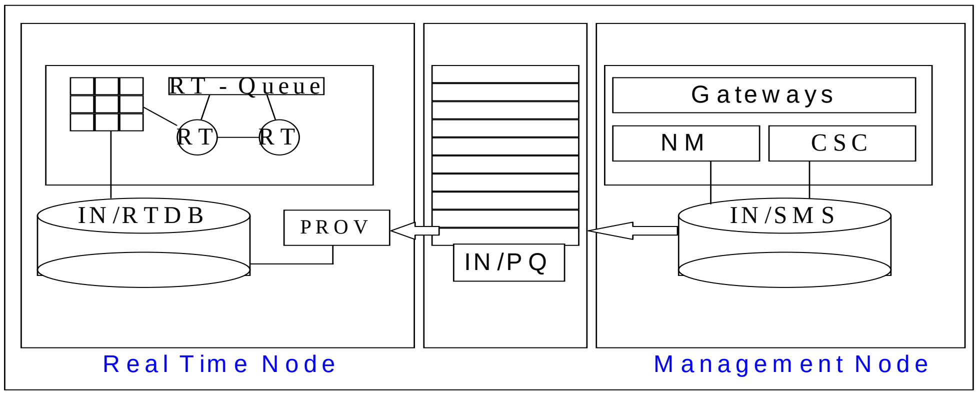

**Figure 5: IN Data Management Layer**

Figure 5 represents the architecture for the IN data management layer. Oracle can meet the call processing and provisioning requirements. Oracle meets the call processing requirement by configuring the instance to cache the entire real-time (user data) into memory. The application use standard SQL to perform the read (select) and write (insert/update/delete) transactions. The provisioning transactions can be queued using Oracle8 / Advance Queues. O8/AQ is a persistent queue implemented within the database and provides features such as priority, time expiration, queue propagation, etc. It is robust and highly performant, enough that leading Object Management and Transaction Monitor vendors will use it to provide the underlying technology to support their queuing requirements.

Oracle8 also provides a rich and robust set of replication capabilities, which are a good fit for provisioning many types of IN applications. Oracle replication is the ability to replicate data from one database to another, taking into account sets of rules to define, which data, when, and what to do in the eventuality of conflicts. Replication can be peer-to-peer, master to slave, many to many or one to many, real time, or 'snapshot'. Using Oracle replication can greatly simplify many provisioning tasks, especially those where data from one system needs to be copied to a number of outlying systems, for geographic or business reasons, or where changed data at such systems needs to be synchronized back with a central system on a periodic basis. Because it is supported by the underlying technology, it is more robust and much easier to implement than a system in which such functionality needs to be explicitly written by the application provider.

## 3. Case Study

The following text describes in detail how Oracle was configured to meet the near real-time requirements of an IN application. The Home Location Register (HLR) was selected as the case study for this exercise. The Logical Model used for this case study is based on the IN benchmark performed by Oracle in conjunction with various IN application vendors to confirm that the Oracle server meets the stringent IN requirements.

### 3.1 Logical Data Model

The Logical model used represents a simple time dependent B number translation service data structure. The model is shown below.

The primary keys are shown in the above diagram as fields prefixed with a '#' symbol.

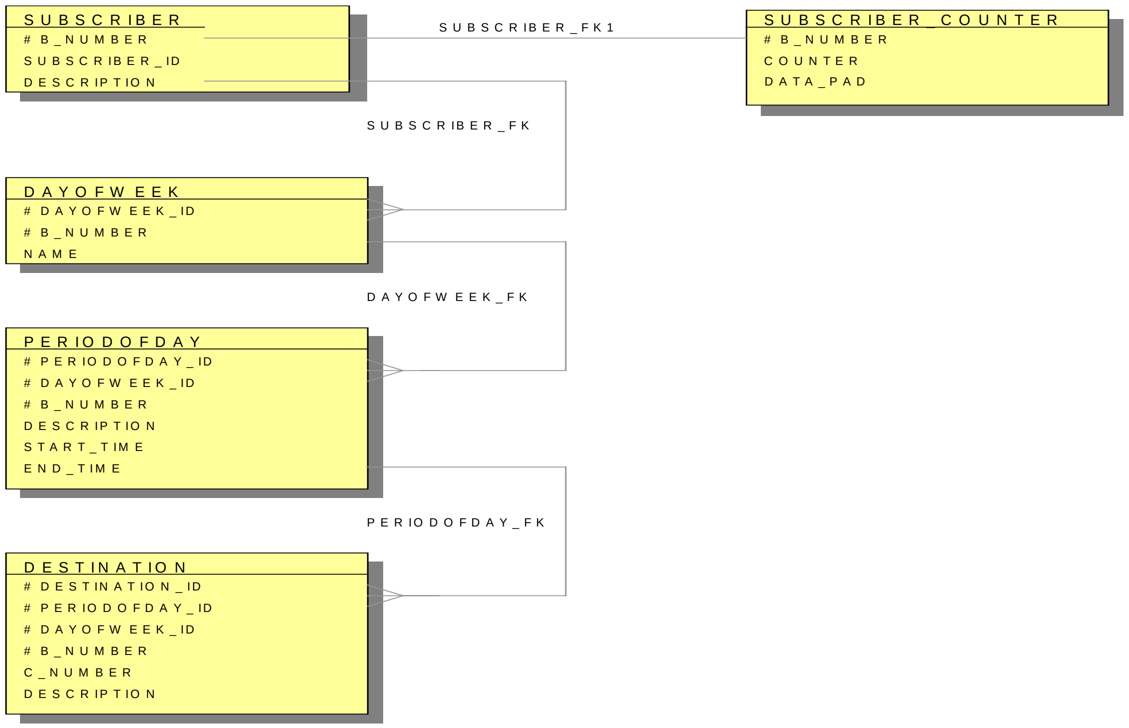

**Figure 6: Logical Data Model**

The model relationship centers around the subscriber record, and since the subscriber's key data is the B-Number, this number represents the number called by the originator. The subscriber has a number of related "DayOfWeek" records. This record type represents a single day in the week. Each “DayOfWeek” record has a related "PeriodOfDay" record that represents sequential periods in a day of arbitrary length. Finally, period records have related "Destination" records that contain a C-Number; this number represents the re-routed or translated destination number. As these related record types are generally used for look up purposes, another subscriber related table is included to allow update processing to take place. The "UpdateCounter" record acts as a test record for just such operations.

The following constraints are placed on the logical data model to simplify the benchmark tests:

- Subscribers are unique
- 7 "DayOfWeek" records exist for each Subscriber record
- 3 "PeriodOfDay" records exist for each DayOfWeek record
- "PeriodOfDay" records may not have overlapping logical time ranges
- "PeriodOfDay" records time range cannot exceed 24 hours
- "PeriodOfDay" records time range cannot carry over to the logical day
- Granularity of time used in period ranges is 1 minute
- 1 Destination record created for each PeriodOfDay record (more are possible but one is used for the purpose of this case study)

For the case study transactions per second are measured as described in the following text. Time is queried before and after a loop of transactions is executed, an average TPS is then calculated from the aggregate transaction time and the number of transactions processed. The benchmark program records the time taken for each transaction and also measures the minimum, maximum, average and the histograms in the interval for 10 ms for each of the transaction types. The average, minimum and maximum are logged every 10 seconds and the histogram every 100 seconds.

## 3.2 Oracle8 Configuration

Following are some key configuration parameters that were used for the proof-of-concept.

### 3.2.1 Hash Clusters

To provide fast access to the real time data, two HASH clusters were created: one to hold the Subscriber table and the second for the "Periodofday" and "Destination" tables. The Oracle internal hashing algorithm was used as the hash function. The two clusters were created in separate table spaces. This enabled us to monitor the disk accesses for each of the tables independently. The following script is an example of merging two table into the cluster to access data faster. This resulting in the join query returning the block containing the entire data set for the subscriber in on I/O.

```
DROP CLUSTER pdest_cluster INCLUDING TABLES;
CREATE cluster pdest_cluster (b_number varchar2(40), dayofweek_id number(1,0) )
             PCTFREE       1
             PCTUSED       40
             INITRANS      2
             MAXTRANS      6
             SIZE          <average row size>
             TABLESPACE    <tablespace>
             HASHKEYS      <number of keys>
   STORAGE
             ( INITIAL 500M  NEXT 500M  PCTINCREASE 0 MINEXTENTS 4
               MAXEXTENTS 100 BUFFER_POOL KEEP);

CREATE TABLE periodofday (
             b_number         varchar2(20) not null,
             dayofweek_id     number(1,0) not null,
             periodofday_id   number(4,0) not null,
             start_time       number(4,0) not null,
             end_time         number(4,0) not null,
             descr            varchar2(400)
      )
      CLUSTER pdest_cluster (b_number, dayofweek_id );
```

```
CREATE TABLE destination (
            b_number        varchar2(40) not null,
            dayofweek_id    number(1,0) not null,
            periodofday_id  number(4,0) not null,
            destination_id  number(2,0) not null,
            c_number        varchar2(40) not null,
            descr           varchar2(400)
    )
    CLUSTER pdest_cluster (b_number, dayofweek_id );

ALTER CLUSTER pdest_cluster CACHE;
```

### 3.2.2 Referential Integrity

Primary and foreign keys were created on all the required tables to maintain the data integrity. It was observed that there was no impact on the real time performance (Selects & Updates) due to these constraints. Hash clusters are good only if all queries use exact key values. If not, the queries will perform a table scan, resulting in unacceptable results. It is recommended the primary and secondary keys be created to support these queries. If possible, place the secondary keys on separate controllers than the data (hash clusters).

```
CREATE UNIQUE INDEX subscriber_pk ON subscriber(b_number)
            PCTFREE         1
            INITRANS        2
            MAXTRANS        6
            TABLESPACE  <index tablespace>
            STORAGE         ( INITIAL 32M  NEXT 32M  PCTINCREASE 0
                              MINEXTENTS 2 MAXEXTENTS 100 BUFFER_POOL KEEP)
NOLOGGING;

CREATE UNIQUE INDEX periodofday_pk ON periodofday(b_number, dayofweek_id, periodofday_id)
            PCTFREE         1
            INITRANS        2
            MAXTRANS        6
            TABLESPACE  <index tablespace>
            STORAGE         ( INITIAL 128M  NEXT 128M  PCTINCREASE 0
                              MINEXTENTS 2 MAXEXTENTS 100 BUFFER_POOL KEEP)
NOLOGGING;

CREATE UNIQUE INDEX destination_pk ON
            destination(b_number, dayofweek_id, periodofday_id, destination_id)
            PCTFREE         1
            INITRANS        2
            MAXTRANS        6
            TABLESPACE  <index tablespace>
            STORAGE         ( INITIAL 128M  NEXT 128M  PCTINCREASE 0
                              MINEXTENTS 2 MAXEXTENTS 100 BUFFER_POOL KEEP)
NOLOGGING;
```

### 3.2.3 Pin Data Into Memory

One of the key criteria for the benchmark is to load the entire real time data into memory. This was achieved by configuring multiple parameters. Oracle8 provides for multiple pools of buffer in the data buffer component of the SGA. The BUFFER_POOL_KEEP was configured for both tests to load the appropriate data into memory.

The STORAGE parameter for the clusters were configured to pin data into buffer pool KEEP. In addition the DB_BLOCK_LRU_LATCHS size was increased from the default (CPUs/2) to maximum allowed ( CPUs * 2 ).

The next step is to load the data into memory. The first command to pin the data into memory is to ALTER the cluster (ALTER CLUSTER <cluster name> CACHE). To load the data into memory the table need to be scanned. The statement 'SELECT * FROM table' does scans the entire table but does not load the entire table into memory. This is because Oracle determines that it is a full table sequential scan and reuses the same block in memory while performing the reads. This can be overcome by using a LIKE clause in the WHERE statement ('SELECT * FROM SUBSCRIBER WHERE SUBSCRIBER LIKE '%'). The table can be loaded 70% faster into memory by using the ANALYZE command. ANALYZE CLUSTER <cluster name> VALIDATE STRUCTURE. This command loaded the entire cluster into memory.

```
ALTER TABLE         <table name>    ENABALE TABLE LOCK;
ANALYZE CLUSTER     <cluster name> VALIDATE STRUCTURE;
ALTER TABLE         <table name>    DISABLE TABLE LOCK;
```

The above script is an example for loading the entire data into memory. As the IN applications do not require table level locking, it is recommend that the table locks be disabled for all tables that store real-time user data. The table level locks have to be enable for the ALTER TABLE command.

The process for pinning data into memory can be made faster by using the DBMS_ROWID procedures. By using this procedure and creating a separate table, multiple queries can be submitted in parallel to scan all the rows within some blocks.

#### 3.2.3.1 Loading data into memory in a live situation

Whilst the benchmark simulated the real-life situation of the database being contained in memory by explicitly loading it in order to get to the steady state situation quicker, it is not always necessary, or advisable, to do this for a live system. There is a trade-off between start up time and response time, since the workload caused by reading a very large table into memory will inevitably interfere with response time, such that the application may be configured to wait until the load into memory is completed, before accepting network requests.

Some users therefore find that it is better to allow the data to migrate over time into memory cache, as requests occur. Whilst initial queries against the database may be slower due to the need for disc access, they also do not have to contend with the workload of loading a potentially very large database into memory. As long as the ramp-up of queries into the database is relatively slow, it is probably a better option not to try and pre-load the database. If however, there is an almost instantaneous peak of access, then the better alternative would probably be to pre-load, or throttle the initial number of queries.

The choice as to which way to configure the application is dependent upon the trade off between the need for rapid start up, the need to meet particular response times if this can only be done through in-memory access, and the size of the database in question.

### 3.2.4 Database Index Configuration-Partitioning & Reverse key Indexes

Many IN systems are characterized by one or two key indexes on the underlying database, against which access is made. For example, IMSI, MSISDN in GSM, and ESN (Electronic Serial Number) in Analog wireless systems . Setup of the database in respect to the indexes can be crucial in ensuring that access is evenly balanced across systems and system components. An unbalanced system can be a key factor in poor performance, and bad design can, in the worst case, lead to an inability to improve performance irrespective of the amount of computing hardware utilized. The Oracle database contains a number of capabilities to improve the ability of the designer to balance access. In particular, partitioning and reverse key indexes are the most frequently used.

Database tables can be partitioned, transparently to the application, to allow placing on separate discs/controllers/systems, and to allow partitions to be split/joined and moved, without needing to amend the application. Partitioning allows particular sets of data to be explicitly allocated to the underlying hardware, in order to better balance use of the underlying hardware and thus throughput. Partitioning may also be used where it is necessary to place a particular subset of data onto particular hardware. Partitioning may be transparent to the Oracle database, (ie implemented by designer the without defining it within the database schema, and it may also be used explicitly, using Oracle's partitioning to control placement. For example, an operator might place all customers who belong to a specific service provider on separate systems (implicit partitioning), and then within those systems, allocate partitions based on ESN. Partitioning also increases availability, because should the data within a partition be unavailable, access to other partitions is unaffected. Partitioning works independently of Reverse Key Indexes.

The Oracle database contains a native reverse-key function, such that whilst access is made against the real key, on disc it is held reversed. This is setup by a simple keyword when the database is configured, and is transparent to any applications accessing the data, so that for example, a key of 1234 would be held internally within the database as 4321. This is invisible to the application, which would make its accesses against the 'real' key of 1234, in this case. The reason that this is often a successful strategy is that it is usually the leading digits of a key which tend to be less random, for example they may contain area code, or operator specific digits, and it is the last few digits that tend to vary. By reversing the key, it is not uncommon for an almost perfectly random spread of data to be created, which will never necessitate rearrangement, other than defining the number of partitions into which it will be split. This can be changed over time, as the database throughput grows, without any changes at all to the underlying application.

In general, the only time that reverse key index for this type of frequently accessed data is not a good idea, is for systems where data belonging to a particular subset of customers, ordered by key, needs to be held on a restrained subset of systems, or for systems where large sequential scans of the underlying data are often made.

### 3.2.5 Disk Configuration

Due to the performance requirements of an IN system, great care should be taken in developing the disk configuration (physical model) of the database. It would be advantageous to have as many controllers and disks as possible, within reasonable limits.

The data and indexes should be kept on separate tables spaces and preferably on different controllers (if possible). The disks should be striped to ensure that there are no disk I/O contentions. In addition, the Redo log files and the archive files should be on different disks and controllers. The archive log file can be based on the Unix file system and both the redo log files and the archive log files should not be striped. If possible, it would be advantageous to have 2 controllers for the redo log files with the odd number files being assigned to disks on controller 1 and the even number files assigned to disks on controller 2. This will facilitate serial writes and reads of the log writer and the archiver process. This will ensure that both these process do not create a contention at the controller or disk level, specially whenever the redo log switches.

Our observation has been that Oracle can meet the high end requirements using two controller with fiber channel access to the disks. Oracle requires a minimum of 3 (preferably 4) controllers for systems with traditional controllers. In case of HP systems, Oracle recommends using two Auto RAID units, one for data and indexes and the other for Redo Log files.

Always use raw devices and asynchronous IO whenever possible. It is observed that the throughput goes up substantially by avoiding the overhead of system buffer (jfs/ufs). Set DBWR_IO_SLAVE and LGWR_IO_SLAVE, depending on the OS support for asynchronous IO.

### 3.2.6 Consistent Read Parameter

It was observed that initially when the system first started up there were no read I/O's on the data disk. After a period of time, it was observed that Oracle8 was performing some read I/O's on the data disk. Oracle keeps copies of multiple blocks in memory to support consistent read. Oracle8 defaults to 10 for the number of copies for the consistent read blocks. This was reduced to 4 by setting the _DB_BLOCK_MAX_CR_DBA parameter. To achieve the required throughput, Oracle's recommendation is that the DB_BLOCK_BUFFER should be nearly twice the size of the hash cluster.

### 3.2.7 Discrete Transactions

The write transactions are faster by using discrete transactions. To enable discrete transactions, the INIT.ORA parameter ENABLE_DISCRETE_TRANSACTION has to be set to true. The discrete transaction is invoked by calling the procedure DBMS_TRANSACTION.BEGIN_DISCRETE_TRANSACTION.

```
EXEC SQL EXECUTE
        BEGIN
                dbms_transaction.begin_discrete_transaction;
        END;
END-EXEC;

EXEC SQL UPDATE .....;
EXEC SQL COMMIT;
```

Please read the manual and review the application before using discrete transactions.

### 3.2.8 Rollback Segments

Oracle8 server tuning guide recommends that N / 4 rollback segments be created where N is the number of concurrent transactions. Please use this formula to create rollback segments.

### 3.2.9 Transparent Application Fail over

Oracle now provides for transparent application failover and work in a client server mode. To test this facility the application was runon a separate node, accessing the database over the network using Net8. The TNSNAMES.ORA on the client node was configured for Transparent Application Fail over. The fail over mode was configured as PRECONNECT, this resulted in the client connecting to both the nodes whenever it connected to the database. The application would switch to the backup node instantaneously whenever it detected that the primary node was down.

The failover was simulated by killing all the Oracle processes on the primary node. The client detected the failure instantaneously and connected to the secondary node. As the FREEZE_DB_FOR_FASTA_INSTANCE_RECOVERY was set to FALSE, all blocks, other than ones being modified were available on the secondary node. It took Oracle approximately 30 seconds to perform the recovery on the modified blocks. The application was able to cater to approximately 40% of the transactions with the required amount of time. It took a couple of minutes to load the entire data into memory on the secondary node. This was done by using the SELECT * FROM <table name> WHERE var1 LIKE '%'. The entire data could be loaded into memory faster by submitting multiple queries using DBMS_ROWID procedures. Oracle8 currently support transparent application failover only for applications developed OCI and can presently roll forward only the SELECT transactions.

```
Ops1.world =
(DESCRIPTION =
  (ADDRESS_LIST =
    (ADDRESS =
      (COMMUNITY = tcp.world)
      (PROTOCOL = TCP)
      (Host = <hostname>)
      (Port = 1521)
    )
  )
  (CONNECT_DATA = (SID = ops1)
  (FAILOVER_MODE = (TYPE = SELECT )(METHOD = PRECONNECT)
  (BACKUP = ops2.world))
  )
 )
```

### 3.2.10 Lock Configuration

Oracle Parallel server uses Hash level locking as well as fine grain locking. These parameters should be calculated based on the transactions for optimum transactions. The GC_FILES_TO_LOCK parameter was set for the benchmark.

### 3.2.11 Provisioning transaction using Oracle8/AQ

The following mechanism was used to prototype the replicating transactions for IN applications between the SCP and the management systems.

- Create an IN queue and an OUT queue on the OPS (IN real time database) nodes.
- Create triggers for insert, updates and deletes on the SMS node to ENQUEUE the transactions to the IN queue on the OPS node. This was done by using database links and making remote procedure calls to ENQUEUE transaction to the real-time node.
- A demon process on the real-time node would wait for a message in the IN QUEUE and apply the transaction to the real-time node.
- Create triggers for insert, updates and deletes on the OPS nodes to ENQUEUE transactions to the OUT queue on the OPS node. Add logic to ensure that only the network transactions are queued back to the SMS node.
- A demon process on the SMS node would wait for a message in the OUT queue and apply the transaction to the SMS database.

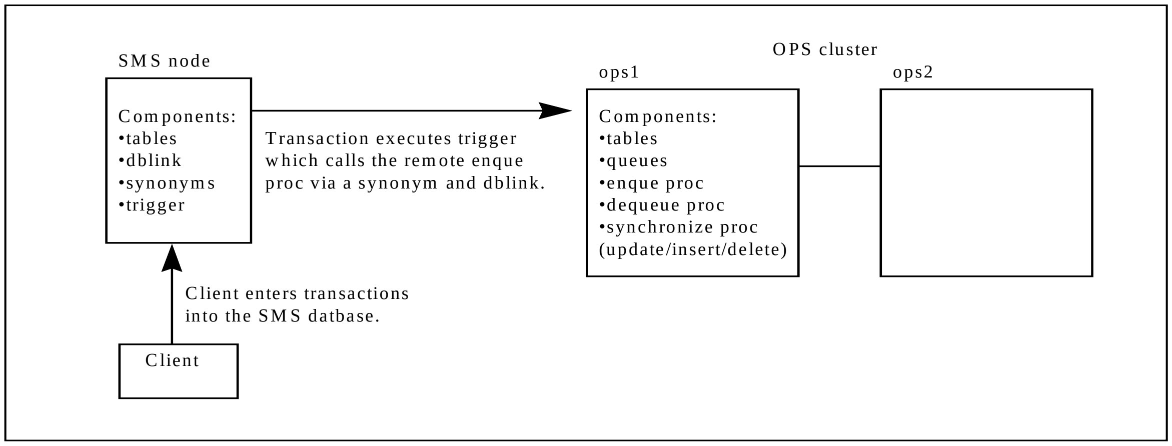
SMS node
Components:
•tables
•dblink
•synonyms
•trigger
Transaction executes trigger which calls the remote enque proc via a synonym and dblink.
OPS cluster
ops1
Components:
•tables
•queues
•enque proc
•dequeue proc
•synchronize proc (update/insert/delete)
ops2
Client enters transactions into the SMS datbase.
Client

**Figure 7: Provisioning Transactions**

The above figure represents the provisioning transaction process. Oracle 8.0.4, now supports queue propagation, thereby eliminating the need to ENQUEUE transaction to the IN queue on the OPS node. Due to the overhead of Oracle8 Replication, it was determined that the transactions be replicated using O8/AQ. Based on the preliminary performance numbers of Oracle 8.1 (pre beta), it will be possible to meet the transaction throughput using Oracle Replication for entry level and mid rage systems.

## 3.3 Comments

Following are the comments based on the results observed during the benchmark:

- Typically, for IN application, the platform vendors have a tendency to log the time results after every transactions for analysis purposes. Based on previous experience Oracle expects a 20% throughput improvement by NOT logging every transaction. Oracle's HLR driver, logs the results every 10 seconds and prints the histogram every 100 seconds.
- The benchmarks drives were developed using Pro*C. Oracle recommends that the designers review using PL/SQL if the transactions require multiple queries. This is to reduce the round trip between the application and the server.
- Discrete transactions were used for updates to reduce the overhead. The discrete transaction is enabled by calling a PL/SQL procedure.
- As the numbers of programs increased the transactions rate per process does not remain the same. This us due the due to the overhead of context switching. The throughput is expected to improve between 5% and 15% by using thread safe facility provided by Oracle. The optimum number of threads for a process is directly related to the number of CPUs in the system.
- The throughput increased by approximately 100% by using single-task applications. Even though this is provided by Oracle, Oracle does not support single-task applications.

Oracle can provide additional benefits in the following area for IN applications:

- Oracle8 / Advance Queuing is a new feature provided with Oracle 8 as part of the database. O8/AQ is a persistent queue implemented in the database with all the standard queuing mechanism such as priority, time expiration, propagation to remote nodes, etc.
- Oracle8 / Replication can assist in the implementing mated pair architecture.
- Oracle Parallel Server can be used for High Availability/Fail over. Oracle has been able to configure OPS for IN applications in test conditions to fail over within the specified time limit.

## 3.4 Common INIT.ORA parameters

Following are some of the INIT.ORA parameters for review purposes. These parameters could be used as baseline for configuring IN applications/platforms.

| | |
|---|---|
| DB_NAME | ops1 |
| CONTROL_FILES | ...../controlfile1 |
| | ...../controlfile2 |
| | ..../controlfile3 |
| db_block_max_dirty_target | 100 |
| DB_BLOCK_CHECKPOINT_BATCH | 100 |
| OPTIMIZER_MODE | CHOOSE |
| DB_BLOCK_SIZE | 4096 <recommend 4K blocks> |
| PARALLEL_SERVER | TRUE <for OPS only> |
| FREEZE_DB_FOR_FAST_INSTANCE_RECOVERY | FALSE <for OPS only> |
| CPU_COUNT | <variable> |
| _DB_BLOCK_MAX_CR_DBA | 4 |
| SPIN_COUNT | 10000 <dependent on hardware> |
| CLOSE_CACHED_OPEN_CURSORS | TRUE |
| CURSOR_SPACE_FOR_TIME | TRUE |
| OPEN_CURSORS | 100 |
| DB_FILE_SIMULTANEOUS_WRITES | 40 <variable> |
| DISK_ASYNCH_IO | TRUE <always use ASYNCH IO> |
| DBWR_IO_SLAVES | 4 <OS dependent> |
| LGWR_IO_SLAVES | 4 <OS dependent> |
| LOG_SIMULTANEOUS_COPIES | 8 |
| DB_BLOCK_LRU_LATCHES | 8 <based on # of CPUs> |
| SERIAL_REUSE | ALL |
| BUFFER_POOL_KEEP | 660000 |
| CACHE_SIZE_THRESHOLD | 660000 |
| DB_BLOCK_BUFFERS | 730000 |
| SHARED_POOL_SIZE | 25M |
| SHARED_POOL_RESERVED_SIZE | 6M |
| SHARED_POOL_RESERVED_MIN_ALLOC | 2M |
| PRE_PAGE_SGA | TRUE |
| DISCRETE_TRANSACTIONS_ENABLED | TRUE |
| COMPATIBLE | 8.0.4 |
| DB_FILES | 80 |
| DB_FILE_MULTIBLOCK_READ_COUNT | 16 |
| DB_BLOCK_CHECKPOINT_BATCH | 32 |
| log_checkpoint_interval | 99999999999 |
| log_checkpoint_timeout | 0 |
| PROCESSES | 50 |
| DML_LOCKS | 50 |
| log_buffer | 2097152 |
| sequence_cache_entries | 10 |
| sequence_cache_hash_buckets | 10 |
| timed_statistics | TRUE |
| max_dump_file_size | 10240 |
| LOG_ARCHIVE_BUFFER_SIZE | 2048 |
| LOG_ARCHIVE_FORMAT | arch%s.arc |
| LOG_ARCHIVE_START | TRUE |
| GLOBAL_NAMES | TRUE |