

Implementing ILM using Oracle Database 11g

An Oracle White Paper
June 2007

Implementing ILM using Oracle Database 11g

Introduction	4
What is ILM?	4
ILM or HSM.....	5
Why Oracle Database 11g for ILM.....	5
The Benefits of using a Database	5
Structured Data.....	6
Email, Documents, Images and Other Types of Data	6
Implementing ILM Using Oracle Database 11g.....	6
Step 1 – Define the Data Classes	7
Partitioning.....	7
Step 2 – Create Storage Tiers for the Data Classes	8
Step 3 – Create Data Access and Migration Policies.....	10
Controlling Access to Data.....	10
Security at the Database Level	11
Controlling the Data visible to a User	11
Restricting Access to Historical Data	12
Enterprise Identity & Security Control	13
Moving Data using Partitioning.....	13
Regulatory Compliance.....	14
Step 4 – Define and Enforce Compliance Policies.....	15
Data Retention.....	15
Immutability.....	15
Digital Signatures	16
Data Privacy.....	17
Encryption	17
Auditing Access	18
Standard Auditing.....	18
Fine-Grained Auditing.....	18
Expiration – The Benefits of the Online Archive.....	19
Archiving using XML.....	20
Creating a Custom Archive.....	22
Removing the Data.....	22
Recording all data Changes	22
ILM Assistant.....	23
More Oracle Database 11g Features for ILM	25
Storage Management.....	25
Using Inexpensive Disks.....	26

Automatic Storage Management (ASM)	26
Data Movement Mechanisms & Utilities	27
Relocating Datafiles and Tables	27
Moving Data using Online Redefinition.....	27
Moving Information Between Databases	27
Data Pump.....	28
Transportable Tablespaces	28
Single Partition Tablespaces.....	28
Streams	29
Preventing Data Changes	29
Read-Only Tablespaces	29
Read-Only Tables.....	29
Write-Once Devices.....	30
Storage Reduction Techniques	30
Compression and Removing Unused Space	30
Consolidating Data.....	31
Protecting the Data.....	31
Protection from Human Error and Hardware Failure using Flashback.....	32
Protection from Corruption	32
Protection from Site Failure	33
Real Application Clusters (RAC)	33
The Grid and Information Lifecycle Management	33
Conclusion.....	33

Implementing ILM using Oracle Database 11g

INTRODUCTION

Although most organizations have long regarded their stores of data as one of their most valuable corporate asset, how this data was managed and maintained varies enormously. Originally, data was used to help achieve operational goals, run the business and help identify the future direction and success of the company.

Around the world, a number of regulatory requirements, such as Sarbanes-Oxley, HIPAA and DOD5015.2-STD in the US and the European Data Privacy Directive are changing how and why data is being retained, as they are now requiring organizations to retain and control information for very long periods. Consequently today there are two additional objectives IT managers are trying to satisfy: to store vast quantities of data, for the lowest possible cost; and to meet the new regulatory requirements for data retention and protection.

Oracle Database 11g already contain a rich feature set to support and comply with the ever changing demands required by organizations. This paper discusses in more detail all of the components in Oracle Database 11g, which can be used to build an Information Lifecycle Management (ILM) strategy, for business data held within an Oracle database.

What is ILM?

There is a wide variety of information held in an organization today, for example it could be an email message, a picture, or an order in an Online Transaction Processing System. Therefore, once the type of data being retained has been identified, you already have an understanding of what its evolution and final destiny is likely to be.

The challenge now before all organizations, is to understand how their data evolves and grows, monitor how its usage changes over time, and decide how long it should survive. In addition, the evolving rules and regulations such as Sarbanes-Oxley, place additional constraints on the data that is being retained and some organizations now require that data is deleted when there is no longer a legal requirement to keep it, to avoid expensive e-discovery when the data is requested for a legal matter.

Information Lifecycle Management (ILM) is designed to address these issues, with a combination of processes, policies, software and hardware so that the appropriate technology can be used for each phase of the lifecycle of the data.

ILM or HSM

Before looking in detail at Information Lifecycle Management (ILM), it is helpful to understand the relationship between the concept known as Hierarchical Storage Management (HSM) and ILM.

HSM is a storage subsystem, which was typically used in large enterprises, to address issues such as backup and recovery, archiving data, and storing data on offline or nearline storage. An HSM system will automatically move data to the storage media, which is most appropriate for how it is being accessed. Most storage vendors offer HSM products, but since they are a hardware and or software solution, users must explicitly and correctly, ensure that the HSM systems manage the appropriate files.

Information Lifecycle Management is the next stage beyond the storage-only HSM product and encompasses both software and hardware. No longer do you have to be concerned about issues such as which database files the storage manager can handle, because, the database manages its own files in an ILM environment. Another major benefit is that only large organizations could afford an HSM solution, where as any organization, small or large, can take advantage of Information Lifecycle Management.

WHY ORACLE DATABASE 11g FOR ILM

Oracle Database 11g provides the ideal environment for implementing your ILM solution, because it offers a cost-effective solution, that is secure, transparent to the application and achieves all of this without compromising performance. It also already has the ability to:

- Manage storage
- Use database features beneficial for ILM
- Secure and protect data
- Enforce compliance policies

The Benefits of using a Database

With so many different types of data to store, it can be tempting to hold them in different types of data stores, but managing data in multiple places can result in even more problems.

Oracle Database 11g is capable of storing both structured and unstructured data, which is significantly enhanced in the latest release with SecureFiles. Therefore, using Oracle means that all your data is now much easier to manage, because it is all in one place, instead of being stored in multiple file systems.

Structured Data

The Oracle Database began by storing traditional structured data, such as orders from an OLTP system, or was used to build a Data Warehouse. Today it supports an extensive range of datatypes and is capable of storing any type of structured data.

Email, Documents, Images and Other Types of Data

The Oracle Database can easily store both structured and unstructured data. Recent studies have shown that unstructured data, such as documents, email, spreadsheets can account for up to 80% of the data held within an organization. Traditionally, this data has been stored outside of the database and manually managed, but this scenario can cause many problems, as documents can easily get lost and it can be difficult to control who is viewing them.

Oracle Content Database, which is a database option of Oracle Database Enterprise Edition provides an enterprise-wide content storage and management system, which enables you to store any type of document or image, in an Oracle Database.

Oracle Database 11g saw the introduction of Secure Files which has been specifically designed to address many of the concerns around storing unstructured data within an Oracle Database.

In the past, it may have been questionable as to whether it was efficient to store this data inside the database, rather than use a conventional file based system. Today, the database is the preferred choice because there are performance gains that result from indexing this data, it is easier to manage, and there are all the security features that are available.

Now you can see that if all the information your organization cares about, is held inside an Oracle Database, you can take advantage of the features and functionality provided by the database, to manage and move this data as it evolves during its lifetime.

IMPLEMENTING ILM USING ORACLE DATABASE 11g

The Oracle white paper [*Information Lifecycle Management for Business Data*](#) presents an overview of implementing ILM on Oracle and it explains how data can be described as being; active, less active or historical. Building an Information Management Lifecycle solution, using Oracle Database 11g is quite straightforward, and it can be completed, by following these four simple steps:

1. Define the Data Classes
2. Create Storage Tiers for the Data Classes
3. Create Data Access and Migration Policies
4. Define and Enforce Compliance Policies

Step 1 – Define the Data Classes

There are many ways that data can be classified, the most common type of classification is by age or date, but other types are possible, such as by product or privacy, or a hybrid classification could be used such as by privacy and age.

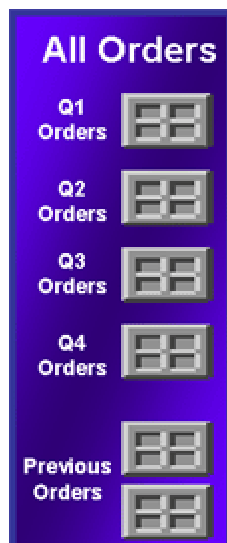
Consider a system where all the orders received by a company are stored. If date is chosen as the class, then when the data is grouped at the row level by their class, which in this example would be the date of the order, all orders for Q1 can be managed as a self contained unit, where as the orders for Q2 would reside in a different class. In Oracle these classes can be implemented by using *partitioning*, and since partitions are completely transparent to the application, the data is physically separated, but the application still sees all of the orders.

Partitioning

Partitioning involves physically placing data according to a data value, and a frequently used technique is to store information, for example, by date. Figure 1 illustrates a scenario where all the orders for Q1, Q2, Q3 and Q4 would be stored in individual partitions and the orders for previous years in other partitions.

Oracle offers several different types of partitioning techniques; such as range, hash, composite and list partitioning, but range partitioning is one of the most frequently used types for ILM. Introduced in Oracle Database 11g is *Ref Partitioning*, which is ideally suited for ILM, or *interval partitioning*, where partitions are automatically created as they are required.

Figure 1 Allocating Data Classes to a Partition



There are a number of benefits to partitioning data, because it provides an easy way to distribute the data across the appropriate storage devices depending on its usage, whilst still keeping the data online and stored on the most cost-effective device. Since partitioning is completely transparent to anyone accessing the data, no applications changes are required, thus it can be implemented at any time and when new partitions are required, they are simply added using the ADD PARTITION clause or can be automatically created by INTERVAL partitioning.

Other benefits of using partitions are that each partition can have its own local index, and when the optimizer uses partition elimination, queries will only access the relevant partitions, instead of all partitions, thus improving query response time.

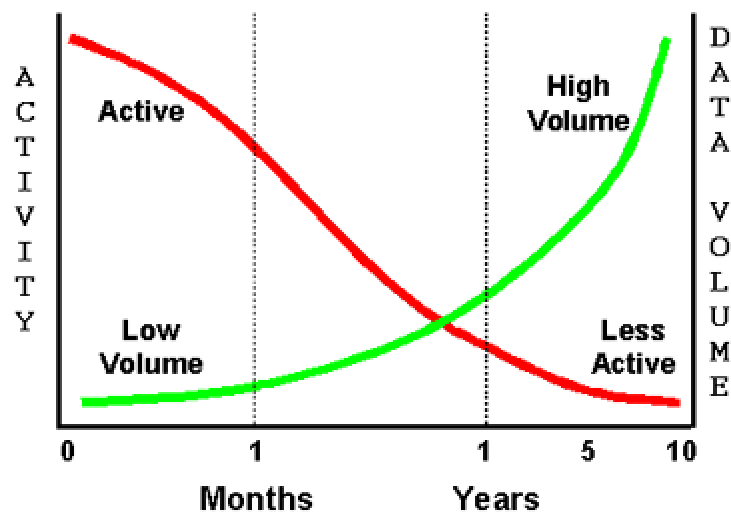
Step 2 – Create Storage Tiers for the Data Classes

Figure 2 illustrates how data is used over a period of time. Using this information, it can then be determined that to retain all this information, several storage tiers will be required to hold all of the data, which also has the benefit of significantly reducing total storage costs.

- High Performance
- Low Cost
- Online Archive
- Offline Archive (optional)

The *high performance* storage tier is where all the important and frequently accessed data would be stored, such as the partition holding our most recent Q1 orders. This would utilize the smaller, faster disks on high performance storage devices.

Figure 2 Data usage over Time



The *low cost* storage tier is where the less frequently accessed data is stored, such as the partitions holding the orders for Q2, Q3 and Q4. This tier would be built using large capacity disks, such as those found in modular storage arrays or the low cost ATA disks, which offer the maximum storage inexpensively.

The *online archive* storage tier is where all the data that is seldom accessed or modified would be stored. It is likely to be extremely large, utilizing low cost storage devices like ATA drives, for a cost that is only slightly higher than storing this information on tape, without the disadvantages that come with archiving data to tape. It could also comprise of read-only devices to guarantee that data placed on this tier is never changed.

The *offline archive* storage tier is an optional tier because it is only used when there is a requirement to remove data from the database and store it in some other format such as XML on a tape.

In the following example, we see how to create these storage tiers in the Oracle Database, by defining tablespaces that reside on the different types of storage devices. A tablespace called ILM_HIGH_PERFORMANCE is created on one of the high performance disks known as /hdsk1.

```
CREATE TABLESPACE "ILM_HIGH_PERFORMANCE"
DATAFILE '/hdsk1/oradata/orcl/ilm_high_perf' SIZE 10G
AUTOEXTEND ON NEXT 1G MAXSIZE UNLIMITED LOGGING
EXTENT MANAGEMENT LOCAL SEGMENT
SPACE MANAGEMENT AUTO ;
```

The low cost storage tier is on a device known as /lcdsk1.

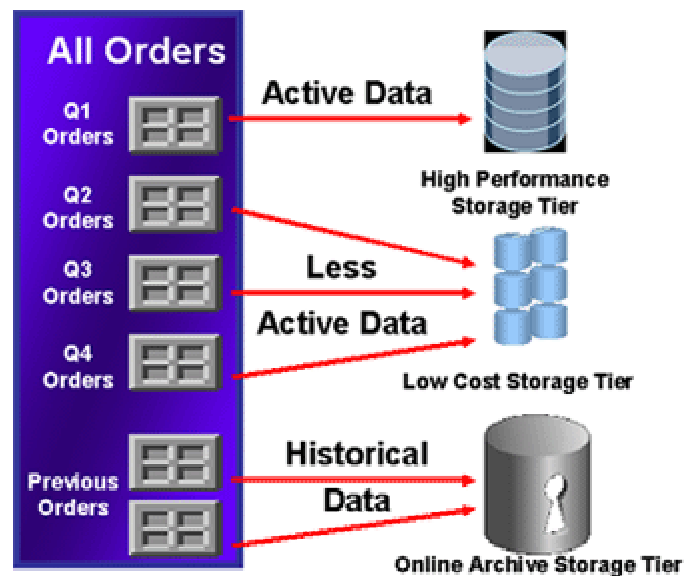
```
CREATE SMALLFILE TABLESPACE "ILM_LOW_COST"
DATAFILE '/lcdsk1/oradata/orcl/ilm_low_cost' SIZE 100G
AUTOEXTEND ON NEXT 10G MAXSIZE UNLIMITED LOGGING
EXTENT MANAGEMENT LOCAL SEGMENT
SPACE MANAGEMENT AUTO ;
```

Since the online archive storage tier will hold vast quantities of data, it is shown here being defined using multiple low cost storage devices.

```
CREATE SMALLFILE TABLESPACE "ILM_ONLINE_ARCHIVE"
DATAFILE '/lcdisk10/oradata/orcl/ilm_oarchive1' SIZE 15T
      AUTOEXTEND ON NEXT 1G MAXSIZE UNLIMITED,
      '/lcdisk11/oradata/orcl/ilm_oarchive2' SIZE 5T
      AUTOEXTEND ON NEXT 1G MAXSIZE UNLIMITED
      LOGGING EXTENT MANAGEMENT LOCAL SEGMENT
SPACE MANAGEMENT AUTO ;
```

Now that the storage tiers have been created, the data classes identified in step 1 will be physically implemented inside the database using partitions. Here we can see how to create our ORDERS tables, and allocate the data classes (partitions) to the appropriate storage tiers (tablespaces)

Figure 3 Allocating Data Classes to Storage Tiers



This approach provides an easy way to distribute the data across the appropriate storage devices depending on its usage, as illustrated in figure 3, whilst still keeping the data online and readily available and stored on the most cost-effective device.

In this example the most recent orders are stored on the high performance storage tier, the orders from the previous three quarters are placed on the low-cost storage tier and everything else is stored on the online archive storage tier.

```
CREATE TABLE orders (  
  prod_id          NUMBER          NOT NULL,  
  cust_id          NUMBER          NOT NULL,  
  time_id          DATE            NOT NULL,  
  channel_id       NUMBER          NOT NULL,  
  promo_id         NUMBER          NOT NULL,  
  quantity_sold    NUMBER(10,2)    NOT NULL,  
  amount_sold      NUMBER(10,2)    NOT NULL)  
--  
-- partitions  
--  
PARTITION BY RANGE (time_id)  
( partition orders_pre_2002 VALUES LESS THAN  
  (TO_DATE('1/1/2003','DD/MM/YYYY'))  
  TABLESPACE ILM_ONLINE_ARCHIVE,  
  partition orders_2003 VALUES LESS THAN  
  (TO_DATE('1/1/2004','DD/MM/YYYY'))  
  TABLESPACE ILM_ONLINE_ARCHIVE ,  
  partition orders_janmar_2004 VALUES LESS THAN  
  (TO_DATE('1/4/2004','DD/MM/YYYY')) TABLESPACE ILM_LOW_COST ,  
  partition orders_aprjun_2004 VALUES LESS THAN  
  (TO_DATE('1/7/2004','DD/MM/YYYY')) TABLESPACE ILM_LOW_COST ,  
  partition orders_julsep_2004 VALUES LESS THAN  
  (TO_DATE('1/10/2004','DD/MM/YYYY')) TABLESPACE ILM_LOW_COST,  
  partition orders_octdec_2004 VALUES LESS THAN  
  (TO_DATE('1/1/2005','DD/MM/YYYY'))  
  TABLESPACE ILM_HIGH_PERFORMANCE );
```

It should also be noted that ASM (Automatic Storage Management), which is described later in this white paper, can also be used to manage the data across the storage tiers.

Step 3 – Create Data Access and Migration Policies

The next step is to ensure that only authorized users may access the data and as the data ages, there are a number of techniques that can be used to migrate the data between the storage tiers.

Controlling Access to Data

The security of your data is another very important part of Information Lifecycle Management because the access rights to the data may change during its lifetime. In addition there may also be legal and regulatory requirements that are imposed upon data when it is held in electronic format, which must be complied with. For example organizations such as the tax authorities may demand that you show that

electronic data is secure from unauthorized access and changes, and can report when changes were made and by whom.

Securing the data is considerably easier when it is held inside an Oracle Database, because it can be secured using database features such as:

- Database Security
- Virtual Private Database
- Views

Security at the Database Level

The database is initially protected by the requirement to specify a valid username and password in order to gain access and view the information. Once a user has logged into the database, additional security can be applied using the SQL GRANT and REVOKE commands, to control access to tables, views, columns, and to restrict which database commands may be executed.

In the example below user ILM_NORMAL is only allowed to view the orders, they cannot create or change them, nor are they allowed to create a view.

```
GRANT select ON orders TO ilm_normal;  
REVOKE update, delete, insert ON orders FROM ilm_normal;  
REVOKE create any view FROM ilm_normal;
```

A further concern is restricting the DBA and other privileged users from accessing application data or preventing an application DBA from manipulating the database and accessing other applications. All these situations can be prevented by using *Oracle Database Vault*, which provides control over who, when, and where, application data can be accessed.

Controlling the Data visible to a User

Since databases contain a wealth and a variety of information, not all data is available to everyone. Specific tables can be secured from unauthorized access, but how do you stop access to specific columns or rows in the table?

One technique is to create a *view* over one of more tables, where the view only contains the columns and rows that the user is allowed to see. This approach works well, but it requires access to the data only via the view. Therefore database privileges must be used to deny access to the data via the table and allow it only via the view.

An alternative approach to using the view is to use *Virtual Private Database (VPD)*, which defines a very fine-grained level of access to the database. Policies are created, where it is specified which rows may be viewed and the columns that are visible. Multiple policies can be defined, so that different users and applications see different views of the data. Since the policies are defined at the database level, they are enforced irrespective of how the data is accessed, be it from SQL or any query tool, such as Oracle Discoverer.

VPD operates by transparently rewriting any query to restrict the users view of the data. Therefore, if a policy is defined which states that a user can only view their own data. If that user issued a `SELECT * FROM table`, VPD would rewrite that query as `SELECT * FROM table WHERE employee = 'username'`. This results in the ability to setup a very secure environment where you can rest assured that no unauthorized access to data will occur.

Restricting Access to Historical Data

Another advantage of using the Oracle Database to retain all of the historical data is that VPD, can be used to specify *policies*, which state exactly which data users can see. A default policy would be defined preventing access to the historical data and another policy created allowing specific users to view the historical data. Therefore, if a policy is defined which states that a user can only see data from the year 2004 onwards. If that user issued a `SELECT * FROM orders`, VPD would rewrite that query as `SELECT * FROM orders WHERE time_id > '31-Dec-2003'`.

Using this approach, the data is still available to those who need access to it, but for the vast majority of users, it is now invisible and therefore is not included or accessed by any of their queries. This is illustrated in the following example.

The first step in defining a VPD policy is to create a function, which specifies that user `ILM_POWER` may view all data, and user `ILM_NORMAL` can only see data from 2004. All other users will not be allowed to view this data.

```
CREATE OR REPLACE FUNCTION ilm_seehist
  (owner IN VARCHAR2, ojname IN VARCHAR2)
  RETURN VARCHAR2 AS con VARCHAR2 (200);
BEGIN
  If SYS_CONTEXT('USERENV','SESSION_USER') = 'ILM_POWER' THEN
    -- see all data
    con:= '1=1';
  ELSIF
    SYS_CONTEXT('USERENV','SESSION_USER') = 'ILM_NORMAL ' THEN
    -- see 2004
    con := 'time_id > "31-Dec-2003"';
  ELSE
    -- no data
    con:= '1=2';
  END IF;
  RETURN (con);
END ilm_seehist;
```

Create the security policy using the package `DBMS_RLS`, which will use the function `ilm_seehist` which was created above

```
BEGIN
DBMS_RLS.ADD_POLICY (object_schema=>'ILM', object_name=>'orders',
policy_name=>'ilm_view_history_data',
function_schema=>'ilm', policy_function=>'ilm_seehist',
sec_relevant_cols=>'time_id');
```

END;

Now we can see the effect of this security policy when different users query the data. When user ILM_POWER queries the data they see all of the rows

```
SQL> connect ilm_power/ilm;
SQL> select count(*) from ilm.orders;
```

```
  COUNT(*)
  916039
```

When user ILM_NORMAL queries the data, they only see the orders created this year

```
SQL> connect ilm_normal/ilm;
SQL> select count(*) from ilm.orders;
  COUNT(*)
    6039
```

Enterprise Identity & Security Control

So far we have only considered access at the database level, but rarely does have a business have just one database. Oracle Identity Management, which is part of Oracle Application Server, provides the capability to perform identity management and security services for the enterprise for web-based applications. Integrated with Oracle Database 11g, it provides a central point of control for users gaining access to information.

For example, using Oracle Single-Sign-on, which is one of the components of Oracle Identity Management, once a user is authenticated, they will have access to all the applications that they are authorized to access. Thus providing one central place to specify which applications a user may execute, and for the user, they only have to login once, to gain access to all the applications they require.

Moving Data using Partitioning

Data not only has to be secure, but it will also have to be moved during its lifetime. Since ILM is typically concerned with vast quantities of data, we need to be able to move volumes, quickly and easily. Incorporating *partitioning* into the database design can facilitate very easy movement of data as its usage changes during its lifetime. When the information in that partition is no longer being regularly accessed, the partition can be moved with the single MOVE PARTITION command, to the inexpensive ATA disks.

In the following example, a new partition is created on the high performance storage tier to hold the orders for January to March 2005. The orders for October to December 2004 that were on the high performance storage tier are moved to the low cost storage tier. Figure 4 shows the final outcome after these operations.

```
ALTER TABLE orders ADD PARTITION orders_janmar_2005
VALUES LESS THAN (TO_DATE('1/4/2005','DD/MM/YYYY'))
TABLESPACE ilm_high_performance UPDATE INDEXES
```

```
ALTER TABLE orders MOVE PARTITION orders_octdec_2004
TABLESPACE ilm_low_cost UPDATE INDEXES;
```

Figure 4 Partitions on Different Storage Tiers

ORACLE Enterprise Manager 10g Database Control

Database Instance: orcl > Tables > Edit Table: ILM.ORDERS

Partitions

Select	Partition Name	High Value - TIME_ID (DATE)	Tablespace
<input checked="" type="radio"/>	ORDERS_PRE_2002	01/01/2003	ILM_ONLINE_ARCHIVE
<input type="radio"/>	ORDERS_2003	01/01/2004	ILM_ONLINE_ARCHIVE
<input type="radio"/>	ORDERS_JANMAR_2004	01/04/2004	ILM_LOW_COST
<input type="radio"/>	ORDERS_APRJUN_2004	01/07/2004	ILM_LOW_COST
<input type="radio"/>	ORDERS_JULSEP_2004	01/10/2004	ILM_LOW_COST
<input type="radio"/>	ORDERS_OCTDEC_2004	01/01/2005	ILM_HIGH_PERFORMANCE

TIP Enter the keyword MAXVALUE as a high value to create a partition to hold all rows whose value for this partitioning column is equal to, or exceeds the highest value defined by all other partitions for this column.

From Oracle Database 11g, the data can be moved even if users are accessing the data in the partition, without disrupting any users. This is achieved by using the online redefinition facility via the PL/SQL package DBMS_REDEFINITION.

Partitioning can also be used to move data between partitioned and non-partitioned tables using the EXCHANGE PARTITION command. Therefore, data could be collected into an ordinary table and then moved into a partitioned table when it is no longer regularly accessed or it can be removed from a partitioned table and placed in an ordinary table.

Using partitioning does not mean that data can only be moved as a partition. If the ROW MOVEMENT clause is enabled for the partition, then rows can automatically be moved from one partition to another. For example, suppose our customers table is partitioned by active and inactive customers. When the column which states that the customer has not placed an order in the last 12 months is updated, this would result in this customer's record automatically being moved to the partition on the low-cost storage device where we keep inactive customers.

REGULATORY COMPLIANCE

The regulatory requirements such as Sarbanes-Oxley, HIPAA, DOD5015.2-STD in the US and the European Data Privacy Directive in the European Union are playing a key role in the long-term retention of data because they are imposing

strict rules on how data is held. Now organizations have to protect against unauthorized changes and possibly show details of every change ever made to a record.

We have already seen how Oracle Database 11g, already contains a number of features which will enable an organization to comply with the new regulations and these features are described in the Oracle white paper, [Applying Oracle Security Technologies for Regulatory Compliance](#). The white paper, [Best Practices for California SB1386](#), illustrates how the features in the Oracle Database can be used to comply with a specific regulation. e.g. using table and column privileges it is possible to specify who can create a row and restrict who can update and delete those rows. Later in the life of the data, read-only tablespaces, or introduced in Oracle Database 11g, read-only tables, ensure that the data will never change. Database views or VPD can be used to restrict the data a user can see and database auditing records all data access. However, for some data, even more security is required. These features are described in more detail in the Oracle white paper, [Privacy Protection in Oracle Database 10g](#).

Step 4 – Define and Enforce Compliance Policies

If this database is being used to show compliance with the numerous regulations around the globe, the final step is to define compliance policies in the areas of:

- Data Retention
- Immutability
- Privacy
- Auditing
- Expiration

Data Retention

Some of the regulations stipulate how long records must be retained. Within the database, deletion or modification of records can be prevented using database security which was described previously. In the example shown below user ILM_NORMAL is not allowed to update or delete orders but they can insert them.

```
REVOKE update, delete ON orders FROM ilm_normal;
```

When it is time to remove the data, a privileged user would execute the task and to verify that data was removed only by authorized users, auditing which is described later will record all the changes that have been made and attempted.

Immutability

The security offered by database systems was initially seen to be adequate, but some of the new regulations require that you can prove that the data has not been changed by anyone. With some power users holding the ability to access the

database at the physical level, one of the best techniques for proving that data has never be altered is to use *cryptographic* or *digital signatures*.

Digital Signatures

Oracle Database 11g can generate a cryptographic or digital signature using the package DBMS_CRYPTO. A column, record or query result set can be input to the package and the resulting signature can be stored with the data or published to a neutral third party. At anytime, the signature can be generated and compared with the original signature to prove that the data has not been altered in any way.

The following PL/SQL example illustrates how using the DBMS_CRYPTO package, we can encrypt and decrypt the credit card number that is held in our database that is used to pay for the orders received into the system.

```
DECLARE
cc_number VARCHAR2(16) := '4567123443215678';
cc_raw RAW(128) :=
UTL_RAW.CAST_TO_RAW(CONVERT(cc_number,'AL32UTF8','US7ASCII'));

key_string VARCHAR2(8) := 'ilm7demo';
raw_key RAW(128) :=
UTL_RAW.CAST_TO_RAW(CONVERT(key_string,'AL32UTF8','US7ASCII'));

encrypted_cc RAW(2048);
encrypted_string VARCHAR2(2048);
decrypted_cc RAW(2048);
decrypted_string VARCHAR2(2048);

BEGIN
dbms_output.put_line('Credit Card Number : ' ||
CONVERT(UTL_RAW.CAST_TO_VARCHAR2(cc_raw),'US7ASCII','AL32UTF8'));

encrypted_cc := dbms_crypto.Encrypt(
src => cc_raw,      typ => DBMS_CRYPTO.DES_CBC_PKCS5,
key => raw_key);

dbms_output.put_line('Encrypted Credit card Number : ' ||
rawtohex(UTL_RAW.CAST_TO_RAW(encrypted_cc)));

decrypted_cc := dbms_crypto.Decrypt(
src => encrypted_cc,  typ => DBMS_CRYPTO.DES_CBC_PKCS5,
key => raw_key);

decrypted_string :=
CONVERT(UTL_RAW.CAST_TO_VARCHAR2(decrypted_cc),'US7ASCII','AL32U
TF8');
dbms_output.put_line('Decrypted Credit Card Number : ' || decrypted_string);
END;
```

Below we can see the output from this PL/SQL procedure with the original credit card number, its encrypted format and the results of decryption.

Credit Card Number : 4567123443215678

Encrypted Credit card Number :
 334441423444433532353134383644323339423344454638323839393146423639434134
 323543343445363742453030
 Decrypted Credit Card Number : 4567123443215678

In Oracle Database 11g it is also possible to generate a digital signature for the results of a query to prove that no records that comprise the query set have been tampered with using the package DBMS_SQLHASH as illustrated below in Figure 5.

Figure 5 Generating & Comparing Digital Signatures in the ILM Assistant

The screenshot shows the Oracle ILM Assistant interface. The main content area is titled 'Signed Result Sets' and includes a 'New Signed Result Set' button. Below this is a filter options section and a table with the following data:

Signed Result Set Name	Cryptographic Signature Location	Hash Algorithm	Query Text	Generation Date		Last Comparison Date	
CC Transactions	in Fire Proof Safe in Data Centre	HASH_MD5	SELECT tran_dt, description, t...	26-Sep-2006 15:28	Compare	26-Sep-2006 18:00	
Sales by Customer	ON CD #3256	HASH_SH1	SELECT cust_id, SUM (Amount_sol ...		Generate		
2004 Orders	Fire Proof Safe	HASH_MD5	SELECT SUM(amount_sold) from s ...	26-Sep-2006 15:50	Compare		

At the bottom right of the table area, it says '1 - 3'.

Data Privacy

There are a number of methods within Oracle that can be used to ensure data privacy, but the most effective is to create security policies via VPD. This technique was already illustrated in the [VPD](#) example in the section *restricting access to historical data*.

There is no limit to the number of security policies which may be defined, and they operate entirely transparently of the application, therefore no application changes are required. Another important benefit of using this approach is that because the security policies are enforced at the database level, they cannot be overridden.

Encryption

Another level of privacy that is available is the ability to encrypt the data and Oracle Database 11g provides the capability to directly encrypt columns in a table or a tablespace. Therefore, the only method of viewing the encrypted data is via a SQL query, which automatically decrypts the information. When used in conjunction with security policies, it ensures that only authorized persons may view the data.

In the example shown below a table called *credit_cards* is defined where the column *cc_no* which holds the credit card number, is encrypted in the database.

```
CREATE TABLE credit_cards
(cust_id NUMBER,
```

```
cc_name VARCHAR2(30),
cc_no   VARCHAR2(20) ENCRYPT );
```

A tablespace is encrypted when it is created, by specifying the ENCRYPTION USING clause on the CREATE TABLESPACE.

Auditing Access

For business or compliance reasons, you may be required to not only secure your data, but also keep an accurate record of access to it during its lifetime. For example, some authorities demand that for all electronic records you must be able to tell them when it was created, and whenever it was changed and by whom.

Since data can be created and changed using a variety of different methods such as through an application or directly using SQL*Plus, auditing in the database should be enabled, because then, all actions upon the data will be recorded in the audit trail. Through auditing you can monitor any event at the schema, row, statement or at the content level of a column through fine-grained access controls. Even the privileged SYS user cannot escape auditing provided the parameter AUDIT_SYS_OPERATIONS has been set.

Two types of auditing are available in the Oracle Database:

- Standard Auditing
- Fine-Grained Auditing

In Oracle Database 11g there is a view DBA_COMMON_AUDIT_TRAIL which can be queried to see records from both audit trails.

Standard Auditing

Standard auditing monitors activity on statements, privileges and objects. Setting the database initialization parameter AUDIT_TRAIL enables standard Auditing. In the example below, we can see that auditing has been enabled for all insert, update and delete statements on the customers table.

```
SQL> AUDIT insert,update,delete ON ilm.customers BY ACCESS;
```

A query of the audit trail shows that user ILM updated a customers record.

```
SQL> SELECT username, timestamp ,returncode, sql_text FROM dba_audit_trail;
```

<u>USERNAME</u>	<u>TIMESTAMP</u>	<u>RETURNCODE</u>	<u>SQL TEXT</u>
ILM	20-MAY-05	0	update customers set cust_email='velma@abd.com' where cust_id =55

Fine-Grained Auditing

Fine-Grained auditing goes further than standard auditing, because it allows audit policies to be specified, where detailed audit conditions can be defined. For example, with standard auditing, all changes to the *customers* table would be recorded. However, with fine-grained auditing, this could be extended to record

anyone who views or changes a customer record where the credit limit is more than \$10,000, as illustrated below.

An audit record will automatically be written to DBA_FGA_AUDIT_TRAIL table or an event handler can be called, which performs any action, such as send a message to a pager.

```
BEGIN
  dbms_fga.add_policy (
    object_schema => 'ILM',
    object_name   => 'CUSTOMERS',
    policy_name   => 'CUST_LIMIT_CHANGE',
    audit_column  => 'cust_credit_limit',
    audit_condition => ' cust_credit_limit > 10000 ' ;
    statement_types => 'INSERT, UPDATE, DELETE, SELECT'
  );
END;
```

Now when a query looks at a customer record whose credit limit exceeds \$10,000 a record is written to the fine-grained audit trail, as illustrated below.

```
SELECT cust_first_name, cust_last_name, cust_credit_limit from customers where
cust_id =55;
```

The contents of the audit log can be viewed by querying the DBA_FGA_AUDIT_TRAIL as illustrated below.

```
select timestamp, db_user, sql_text from dba_fga_audit_trail where
object_name='CUSTOMERS';
```

```
TIMESTAMP DB_USER    SQL_TEXT
```

```
20-MAY-05    ILM
```

```
SELECT cust_first_name, cust_last_name, cust_credit_limit from customers where
cust_id =55
```

Expiration – The Benefits of the Online Archive

With so much data being retained, the question is often asked, should it be kept online or archived? Tape has always been a popular archiving medium because it is very cheap and can store vast quantities of information cheaply. However, moving data to tape has its own set of problems.

What happens when the data on tape is needed? First the tape has to be found and hopefully it is still readable. The information on the tape must be restored, and provided the data format hasn't changed, the requested report data can be generated. If the data has been archived to tape for many years, then development time may also be needed to write a program to reload the data into the database from the tape archive. This could prove expensive, and time-consuming, especially if the data is extremely old and in a different format. All of this takes time and not an inconsiderable effort on behalf of the operations department who have to manage this service. Another constraint is that some of the regulations stipulate that you must respond to requests for information within a very limited time period.

In the past restoration of archived data was quite a rare event, but today that is not the case. Data is often requested for legal matters going back many years and if that data is available, the organization must produce it in court. If this data was archived many years ago, the cost of performing this e-discovery could prove extremely expensive.

Today it is no longer necessary to archive that data to tape, instead it can remain in the database, or moved to a central archive database. Irrespective of the chosen approach, the data can reside on low-cost storage devices whose cost per gigabyte is very close to that of tape. There are a number of benefits of keeping all of the data in a database. The most important is that the data will always be immediately available and in the current format.

Figure 6 Archive Data to a Central Archive Database

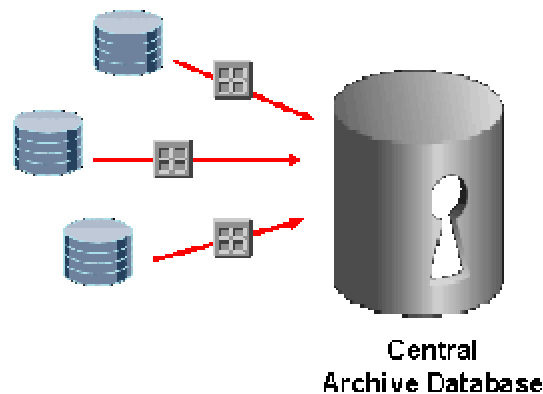


Figure 6 illustrates the approach of archiving data from the various production systems to a central archive database. Using this approach, data is maintained and secured in one location and backups can be taken as required. Alternatively the information could reside in the production database with

no impact upon the time required to backup the database and the size of the backup. When RMAN is used to backup the database, it will only include in the backup, the data that has changed. Since the historical data is no longer changing, it can be declared read-only and once the data has been backed up, it will not be backed up again. Therefore the backup time and storage requirements are the same when compared to as if the data had been archived from the database. i.e. the data is written to tape once and the data is stored on one set of tapes in both cases

Another important factor to consider is how the data is to be physically removed from the database. If the data is partitioned then data removal occurs very quickly. But if the data is retained in the database, then this is another potentially time-consuming maintenance task that is no longer required.

Therefore with the cost of storing the information online rapidly approaching that of tape. A number of important problems can be easily resolved if the information is retained in the database.

Archiving using XML

If it is decided that data eventually has to be archived then it is wise to assume that any data removed from the database, may have to be inserted back into the database at some time in the future. Therefore a format must be chosen that ensures this is possible and a popular solution, is to store the data in XML format,

because it provides a way to describe the data completely independently of any software or hardware. This is an important issue because the format of the data could change over time. e.g. a different datatype could be used for a column, or new columns might be added to the record.

When stored in XML, applications can be written to manage and access the data, so that it can always be viewed, irrespective of how it was originally stored. This technique provides access to legacy data, without the need to maintain applications that require modification every time the data formats change.

Oracle Database 11g includes Oracle XML DB, which provides procedures and functions to extract data into XML format. If your data has been archived such that its format describes its content. Then it is possible to determine the difference with the current format and load it back into the database accordingly.

In the example shown below, all the orders prior to 1995 are extracted into an XML format using the PL/SQL package DBMS_XMLGEN, which is part of Oracle Database 10g.

```
CREATE TABLE temp_clob (result CLOB);

DECLARE
  qryCtx DBMS_XMLGEN.ctxHandle;
  result CLOB;
BEGIN

  -- Select all Orders prior to 1996
  qryCtx := DBMS_XMLGEN.newContext
    ('SELECT * from orders WHERE time_id < "1-Jan-96 " ');

  -- set row header to be ORDERS prior to 1996
  DBMS_XMLGEN.setRowTag(qryCtx, 'ORDERS_PRIOR_1996');

  -- now get result
  result := DBMS_XMLGEN.getxml(qryCtx);
  insert into TEMP_CLOB values(result);
  dbms_xmlgen.closeContext(qryCtx);
END;
```

An alternative method is to embed the XML functions that are available in Oracle Database 11g in your SQL statement, as illustrated below, to generate a customized XML layout.

```
SELECT XMLELEMENT("Orders",
  XMLElement ("timeid", time_id ),
  XMLElement ("prodid", prod_id ),
  XMLElement ("Customer", cust_id ),
  XMLElement ("amount", amount_sold) ) AS "RESULT"
FROM orders WHERE time_id < '1-Jan-96 ';
```

Creating a Custom Archive

Data Archiving is not always a simple matter of extracting data from a single table. For example, suppose an organization keeps all orders for 7 years and after that time they are archived to tape. An order could write records to multiple tables, such as ORDER_HEADER, ORDER_LINES and DELIVERY. Therefore when an order is archived from the database all the related data for an order must be removed at the same time.

This problem can be resolved by writing a custom archive program, but that could be a time-consuming and expensive process. The alternative to writing your own archive program is to generate one using Oracle Warehouse Builder (OWB) which can be used against any database or flat file system.

OWB is a GUI based tool where you specify the all the sources, outputs and transformations, and it generates all the code required to perform this task. This configuration can then be saved and the next time data has to be archived, it can easily be modified if needed, to accommodate any changes to the database, and then a new custom program is quickly generated by OWB.

Removing the Data

There may come a time during the lifetime of the data that it must physically be removed from the database because it is no longer needed, has been moved to another database or regulatory requirements stipulate how much data must be retained.

When data has to be physically removed from the database, one should not underestimate the time this will take if the data has to be removed using conventional SQL delete statements.

The quickest way to remove a set of data from the database is to keep it in its own partition. A simple call to DROP PARTITION will remove all the rows in the table immediately and with the guarantee that all data has been deleted. Here we see how to remove the all the orders held prior to 2002

```
ALTER TABLE orders DROP PARTITION orders_pre_2002;
```

Recording all data Changes

Some regulations state that you must be able to show all the changes that have been made to a record during its lifetime. Previously, this information would have to be maintained within the application, but Oracle Database 11g introduced *Flashback Data Archive*, as part of the Total Recall option, which keeps all changes that have been made to a record in the flashback data archive. Any number of archive files can be specified and assigned to one or more tables. It is recommended that it is used on non-partitioned tables. Below a single flashback archive is created, where data is retained for 5 years for the table orders_online.

```
CREATE FLASHBACK ARCHIVE orders_archive  
TABLESPACE orders_arch RETENTION 5 YEAR;
```

ALTER TABLE orders_online FLASHBACK ARCHIVE orders_archive;

ILM ASSISTANT

This white paper has illustrated the numerous features within the Oracle Database that can be used to build and manage your ILM environment. However, if you prefer to use a GUI tool to perform many of these tasks, then the Oracle ILM Assistant is available. Figure 7 shows the first screen when the ILM Assistant is launched which shows the outstanding tasks that should be performed.

Figure 7 ILM Assistant – Initial Screen



Using the ILM Assistant it is possible to create lifecycle definitions, as illustrated in Figure 8, which are assigned to tables in the database. Lifecycles are described in terms of the *stages*, where data resides during its lifetime.

Figure 8 Lifecycle Definition

Stage:	Current Qtr Orders	Remaining Year Orders	Last Years Orders	Old Orders	Orders End
Tier:	High Performance	Low Cost	Low Cost	Online Archive	
Attributes:	None	None	Compress	Compress	
Retention:	1 Quarter	6 Months	1 Year	2 Years	
Time:	Most Recent Data → → → Oldest Data				

In this example, the data resides on the High Performance tier for one quarter, then, it is move to the Low Cost tier where it remains for 6 months. After 9 months it is then compressed and finally after 19 months it is moved to the online archive tier where it resides for another 2 years.

Using this lifecycle definition, the ILM Assistant advises when it is time to move, archive or delete data, as shown by the calendar. It will also illustrate the storage and cost savings associated with moving the data, as shown in Figure 9.

Figure 9 Lifecycle Events

<input type="checkbox"/> All	Recommended Action	Partition Name	Current Tier	Recommended Tier	Cost Savings	Table Owner	Table Name	Event Date	Event Details
<input type="checkbox"/>	MOVE PARTITION	P20070201_20070228	High Performance	Low Cost	\$4,028	SYSADM	PS_GP_RSLT_ACUM	Past	
<input type="checkbox"/>	MOVE PARTITION	P20070301_20070331	High Performance	Low Cost	\$4,028	SYSADM	PS_GP_RSLT_ACUM	2007/07/01	
<input type="checkbox"/>	COMPRESS	P20070201_20070228		Low Cost	\$759	SYSADM	PS_GP_RSLT_ACUM	Past	
<input type="checkbox"/>	COMPRESS	P20070301_20070331		Low Cost	\$759	SYSADM	PS_GP_RSLT_ACUM	2007/07/01	
<input type="checkbox"/>	PURGE	P19990101_19990131	Online Archive		\$107	SYSADM	PS_GP_RSLT_ACUM	Past	
<input type="checkbox"/>	PURGE	P19990201_19990228	Online Archive		\$107	SYSADM	PS_GP_RSLT_ACUM	Past	

Initially, the ILM Assistant can manage only partitioned tables. For non-partitioned tables, the ILM Assistant will generate a script to show how the table could be partitioned, and it also provides the capability to simulate partitioning on a table to view the actions that would arise, if this table were partitioned, as illustrated in Figure 10.

Figure 10 Lifecycle Tables

Table Owner	Table Name	Storage Size (GB)	Data Reads	Data Writes	Lifecycle Definition	Lifecycle Status	Table Partitioning	Cost Savings	Partition Map
SYSADM	PS_GP_RSLT_ACUM	7,014	19761	0	Employee Activity	Simulated	Simulated	\$474,970	
SYSADM	PS_GP_RSLT_ACUM_ILM	6,818	0	0	Employee Activity	Managed	Implemented	\$432,397	
SYSADM	PS_GP_RSLT_PIN_ILM	3,961	0	0	Employee Activity	Managed	Implemented	\$281,720	
PETE	SALES	3,931	0	0		Candidate	Implemented		
SYSADM	PS_GP_RSLT_PIN	3,906	0	0		Candidate	None		

In its initial release, the ILM Assistant will not execute any tasks for the tasks it recommends to be performed, such as migrate data to different storage tiers. Instead, it generates a script of the commands that needs to be executed

To assist with managing compliance issues, the ILM Assistant will show all VPD and FGA policies that have been defined on tables under ILM control. In addition, both Database and FGA audit records can be viewed and digital signatures generated and compared.

The ILM Assistant requires a minimum of Oracle Database 9i and that Oracle Application Express, formerly known as HTML Db is installed into the database. The ILM Assistant is available for download from the [Oracle ILM page on OTN](#).

MORE ORACLE DATABASE 11G FEATURES FOR ILM

One of the significant advantages of using a database as the basis for your ILM strategy is that it already contains a rich set of features that can be used to successfully implement ILM. In Oracle Database 11g the ones that are of specific interest include:

- Partitioning
- Storage Management
- Data Movement Mechanisms
- Security & Preventing Data Changes
- Storage Reduction Techniques
- Real Application Clusters

Here we will discuss those features that are useful to ILM which have not previously been mentioned.

Storage Management

With so much information to retain, possibly for tens of years, organizations need efficient ways to store this data. Therefore a very important consideration for Information Lifecycle Management is the hardware required to retain this data and the processes needed to manage, secure and access it.

An Oracle database comprises of a number of datafiles, which can reside on three basic types of storage products:

- Disk Storage (SCSI, ATA, Cached Disk Array, etc)
- Optical Disc (CD-ROM, WORM, etc.)
- Tape (not for online access)

Most of the production database files usually reside on a disk storage subsystem or RAID (Redundant Array of Independent Disks) and the database backup sets may reside on tapes or disks.

The trade off between these three types of storage products is usually performance (data access time and throughput) vs. cost (dollar per gigabyte). Disk storage has the highest performance both in terms of access time and throughput, but it is relative expensive comparing it to tapes and optical discs. Both tapes and optical discs are cheaper than disk storage in terms of dollar per megabyte, but they both have limitations. The throughput for a tape device is relatively slow, and it is limited to sequential access only. Optical device's random access time is relative slow, but it has much longer shelf life than both the disk storage and tape.

Using Inexpensive Disks

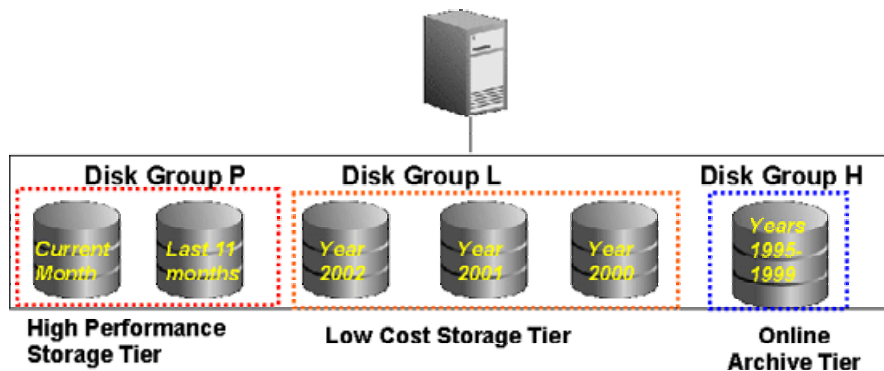
Many storage vendors now offer low cost ATA disk based storage systems in multi-terabyte configurations. The introduction of these devices has a significant impact on ILM because now it is possible to keep vast quantities of data online for a cost that is only marginally higher than using tape storage, without all the problems associated with using tapes. Oracle Database 11g has introduced a number of new features to take best advantage of this shift in disk economics which are described throughout this paper.

Automatic Storage Management (ASM)

Oracle Database 11g introduced Automatic Storage Management (ASM), which provides all the functionality the DBA requires to manage all of these disks. ASM facilitates easy management of large pools of disks, by logically placing disks and files into *disk groups*. Storage devices can now be transparently added or removed from the disk group and all Oracle database files and structures now refer to a disk group rather than a physical location.

Using ASM the DBA can dynamically add and remove disks online, therefore the database can increase in size without having to shut it down. It offers resiliency by mirroring the data across inexpensive storage arrays and it automatically stripes data to ensure that there are no hot spots. An ILM implementation will typically use three storage tiers; high performance, low cost and online archive as illustrated in figure 11. Using ASM each storage tier can be assigned a disk group, thus load balancing within the tier.

Figure 11 Allocating ASM Disk Groups to Storage Tiers



When ASM is used in conjunction with techniques such as data partitioning on a table, the partitions can be spread across all of the disk groups. Later, when it is time to move the data between the disk groups, for example, from the high performance storage tier to the low cost storage tier. This can be achieved using any of the following facilities in the database, move partition, online reorganization or copying a tablespace followed by a rename.

Data Movement Mechanisms & Utilities

There will always come a point during the lifetime of the data when it has to be moved. We have already seen how data can be moved using partition operations, but there are a number of other ways that data can be physically moved within an Oracle database to take advantage of these inexpensive storage devices that are now available. The following techniques can be used together, or in isolation, to physically move the data, without affecting the applications that require it or cause disruption to regular users:

- relocate datafiles and tables
- move tables using online redefinition

Relocating Datafiles and Tables

In Oracle Database 11g, data is placed in a tablespace, which references one or more datafiles where the information is physically stored. An alternative technique to using partitioning is moving the datafiles to a different storage device, using the `ALTER TABLESPACE RENAME DATAFILE` command. This approach requires some forward planning by the database designer to ensure that the datafile being moved does not contain any frequently accessed data. However, it is a simple and quick method for moving data that is no longer needed on a regular basis.

Moving Data using Online Redefinition

Tables can also be moved using the `DBMS_REDEFINITION` package, which allows a table to be moved to another location, and if required, reorganized. This is an online operation, therefore users can continue to access and update the table while it is being relocated.

Moving Information Between Databases

Information may not be retained on one database today, therefore organizations usually have multiple databases and information will invariably move between these databases during its lifetime. For example, orders may be initially input into the OLTP system and then after a few months they may be moved into the data warehouse, which usually resides in a separate database.

There are several fast and efficient methods for moving information between databases:

- Data Pump
- Transportable Tablespaces
- Single Partition Transportable

- Streams

Data Pump

When large amounts of data have to be moved between databases and transportable tablespaces are not appropriate, then Data Pump, which was introduced in Oracle Database 10g can be used. Data Pump Export provides a fast way to unload data to an operating system file. This file is then transferred to the operating system where the destination database resides using standard utilities such as ftp or unix rcp. Then, using Data Pump Import, the data contained in that file is loaded into the database.

Data Pump Export avoids the need to write custom programs because it allows the user to specify which schemas and tables are to be extracted and any data filters can be applied to specify exactly which rows in a table should be contained in the output file.

Transportable Tablespaces

Traditionally, when data has to be moved from one database to another, it is extracted from the source and then inserted into the new database, which can be a very time-consuming process.

Transportable Tablespaces provides a very fast and easy way to move large volumes of data between databases without the need to extract the individual data records. Provided all of the data in the tablespace is self-contained and has no links to data outside of that tablespace, a tablespace can be removed in its entirety from one database and inserted into another. This rapid method works especially well for a partition, which can be placed in its own tablespace and subsequently moved. Thus if the tablespace contained all the orders for 1999, they can be transferred in their entirety to another database. In Oracle Database 11g, the tablespace can be transferred across platforms e.g. from Windows to Solaris.

Another way that transportable tablespaces can be used is to put the transportable tablespace on a CDROM or DVD. This CD can then be mounted and the tablespace added to another database using the Data Pump IMPDP command.

Usually, once the data has been removed from the database, it never needs to be restored, but sometimes this may be required. In this case, the tablespace is simply transported back to the original database as quickly as it was removed.

Single Partition Tablespaces

Introduced in Oracle Database 11g is the ability to use the concept of transportable tablespace, but apply it to one or more partitions. No longer does the database designer have to be concerned in which tablespaces partitions are placed. Now each partition or sub-partition can be moved individually, rather than having to move the entire table or use an exchange partition command.

Streams

Oracle Streams enables sharing of information between databases. With Streams, you can control what is put into the stream, how it is routed, via one or more databases and the events that occur whilst the data is in the stream. Thus Oracle Streams provides a very flexible way to replicate data to other databases.

Any change made to the original database, either DML or DDL, will be captured and replicated at the other database sites and you can use *rules*, to specify exactly what data will be sent. *Transformations* can also be applied to the data should it need to be modified from its original format as it is relocated.

Oracle Streams, in Oracle Database 11g, also provides a mechanism to move tablespaces between databases, without having to define all of the individual steps by using the package DBMS_STREAMS_TABLESPACE_ADM. This package uses transportable tablespaces, Data Pump and the DBMS_FILE_TRANSFER package.

With these capabilities, Oracle Streams provides a mechanism to easily relocate data and maintain that data with changes as they occur, and with the ability to define rules and transformations, there is no longer a need to write and maintain custom extract and load programs.

Preventing Data Changes

There may come a point during the lifetime of the data, or this could be a regulatory requirement, that the data cannot be changed. One approach for ensuring that data never changes is to use Read-Only Tablespaces or Read-Only Tables.

Read-Only Tablespaces

Information inside an Oracle database is stored in tablespaces, and as data matures, you can set a tablespace to be read-only. In the following example, the online archive storage tier is being set to read only.

```
ALTER TABLESPACE ilm_online_archive READ ONLY;
```

Read-Only Tables

Oracle Database 11g introduced the ability to set a table to be read only, which enhances the existing capability to set tablespaces and partitions to be read only.

```
ALTER TABLE online_orders READ ONLY;
```

Once set to read-only, the data cannot be changed, so this approach should be considered for any data that must not be altered under any circumstances. You may be surprised at just how much of your data, falls into this category at some point during its life time, and using this technique you are guaranteed that the information will never be altered. Making a table or tablespace read only is not permanent and it can be switched back to read write at any time.

Since the data cannot change in a read-only tablespace, it is ignored by the Recovery Manager (RMAN) backup utility once it has been backed up. When this technique is applied to historical data, time and system resources will not be spent backing up this data, which can never change.

Write-Once Devices

Another benefit of using read-only tablespaces or tables, is that they are candidates for moving to devices such as a CDROM, DVD or WORM. If the initialization parameter `READ_ONLY_OPEN_DELAYED` is set to `TRUE`, then the Oracle Database will not attempt to access these tablespaces or tables until data is actually requested from them.

Storage Reduction Techniques

With so much data being retained, another important consideration is how much space, stored data is actually using. Reducing storage requirements can also provide some performance improvements for data access as well as backup and recovery operations. In addition there are potential storage costs savings in keeping storage requirements to an absolute minimum. This topic is discussed in the Oracle White Paper [Oracle ILM For Business](#). Within the Oracle Database, there are several techniques that can be used to reduce data storage requirements, these include:

- Compression
- Removing unused space
- Shrinking Datafiles
- Materialized Views

Compression and Removing Unused Space

Once you know that the data is unlikely to change, a table is a candidate for compression. Therefore, when the partition is moved to the low-cost tier from the high performance storage tier, as shown below, it can also be compressed at the same time.

```
ALTER TABLE orders MOVE PARTITION orders_octdec_2004  
TABLESPACE ilm_low_cost COMPRESS;
```

Oracle Database 11g introduced the capability to compress a table. Once this mode has been set, all new data which is inserted into the table will be compressed, but compression does not get applied to any existing data in the table.

```
ALTER TABLE orders COMPRESS;
```

Compression can be disabled using the `NOCOMPRESS` clause, but it will only affect new data being inserted into the table and does not uncompress any previously compressed data.

Another technique for reducing storage requirements is to remove all the unused space that may have accumulated during the lifetime of the data using the online segment shrink option and setting PCTFREE to zero.

```
ALTER TABLE orders SHRINK SPACE;
```

Consolidating Data

When information is being retained, we typically think about holding it in its original form, e.g. the details of the order. An analysis of your data may reveal cases where the data is being held purely for business, rather than regulatory reasons and a summarized view is sufficient for business analysis. e.g. after a period of time you may no longer be interested in all the calls a customer made on a given day, just how much they spent on local, national and international calls.

This data can be summarized into one of more *materialized views*, which can significantly reduce storage requirements. Materialized views are created from the original data source and once created, can be refreshed with only the incremental changes.

In the example below, a materialized view is created, which stores aggregated customer order information on the number and total values of orders made by a customer per month.

```
CREATE MATERIALIZED VIEW Cust_orders_mv
BUILD IMMEDIATE
REFRESH COMPLETE
ENABLE QUERY REWRITE
AS
SELECT o.cust_id, t.month, COUNT(*) AS num_orders,
sum(amount_sold) as total_spend
FROM orders o , time t
WHERE o.time_id=t.time_id
GROUP BY o.cust_id, t.month ;
```

Query Rewrite, when used with materialized views, will automatically and transparently, direct a query to the materialized view for faster access to summarized data, even though the query was directed at the original source data.

Protecting the Data

When vast quantities of data are being retained online, although it is being stored for the lowest possible cost, consideration must also be given to how it can be protected from the following situations:

- Hardware failures using RAC & ASM
- Human errors via Flashback
- Data Corruption using RMAN and Flash Recovery Area
- Complete Disaster Failure using Data Guard

Oracle Database 11g includes the following features, which means that when these events occur, you can quickly recover.

Protection from Human Error and Hardware Failure using Flashback

We are all only human, and it is quite likely that during the lifetime of the database, at some point, someone will make an error that will result in the data becoming invalid, for example, an application error may corrupt the database or an operator error results in a batch job being run twice. This type of data corruption can easily be resolved by using Oracle *Flashback*, which allows the database to be rewound back to a specific point in time using a single command. The database keeps track of all the blocks that have altered, so that a flashback operation, only affects those blocks in the database that have changed.

Protection from Corruption

There are many aspects to consider when designing your ILM strategy, and one that should not be overlooked is Backup and Recovery of the database. An Oracle Database can be backed up using Recovery Manager (RMAN), which provides a mechanism to take complete or incremental backups. The backups can be managed using the Backup Manager in Enterprise Manager Grid Control, where retention policies can be created and applied. When the database needs to be restored, RMAN advises which backup sets are required, based on information in its catalog, and then schedules a restore job.

As databases grow ever larger, the time taken to backup the database, and the storage requirements for those backups also increase. Oracle Database 10g offers an *incrementally updated backup* facility. Instead of regularly taking full backups, the incremental changes to the database are merged into the backup instead. This technique reduces the time and I/O required to perform the backup and significantly reduces the storage requirements for each backup. With reduced demands on available resources, these backups can be held on the inexpensive ATA devices and are always available should a recovery be required.

To protect against more serious corruptions, such as those that may require a complete restore of the database. Oracle Database 11g introduced the *Flash Recovery Area*, which is a configured disk storage area that consolidates all the Oracle database recovery related files.

The Flash Recovery Area is kept online and contains the latest backup of the database. It can be regularly updated with incremental backups so that the backup contains the most recent changes. By keeping the latest backup of the database on disk, recovery is reduced to a matter of minutes and it also permits the use of low cost ATA storage for holding the backup. Thus the combination of both low cost ATA disk storage and Flash Recovery Area delivers a better and faster Oracle database recovery solution than the traditional tape based solution.

Protection from Site Failure

Today most businesses cannot operate for very long without their computer systems, therefore the systems must always be available. Oracle Data Guard provides the ability to maintain both physical and logical standby databases, by automatically transferring data to the standby sites without the need for any user intervention. In the event of a disaster, or planned hardware maintenance, the business can still function by switching to one of the standby databases.

Real Application Clusters (RAC)

With so much data having to be retained and volumes constantly increasing, and users expecting that information is always available, consideration should also be given as to whether a Real Application Cluster is required.

Oracle RAC provides the ability to run all of your applications in a highly available and scalable environment. Where additional resources such as CPU's and storage devices can be configured into the system without disrupting existing users. This highly available environment, means that there is no longer a single point of failure and service can continue on other CPU's in the cluster, should a server fail.

THE GRID AND INFORMATION LIFECYCLE MANAGEMENT

Oracle Database 11g introduces Grid Computing, and Information Lifecycle Management can take advantage of this new architecture. In the Grid, as demand for information grows, additional hardware can be added to Oracle Real Application Clusters databases to support the new processing demands.

To adjust to changing demands from different services using the Grid, components such as databases, processors and storage are integrated but referenced virtually. Whenever there is a change in demand for resources, Grid service brokers can bring in additional resources to cope with the additional demand. Being part of the grid means that it's even easier to adapt as data is changing and moving during its lifetime.

CONCLUSION

Data is an extremely valuable business asset and Information Lifecycle Management enables us to understand our data and how to manage it, thus helping your business remain successful and compliant. ILM encompasses many areas, and a solution is implemented by selecting the appropriate processes, tools and features.

Oracle already has all the various components needed to easily create, store, protect, move and secure your data during its lifetime. Using Oracle Database 11g, a cost effective ILM solution can be built today, which your business can benefit from immediately.



Information LifeCycle Management with Oracle Database 11g
June 2007

Author: Lilian Hobbs
Contributing Authors:

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com

Copyright © 2004, Oracle. All rights reserved.

This document is provided for information purposes only
and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to
any other warranties or conditions, whether expressed orally
or implied in law, including implied warranties and conditions of
merchantability or fitness for a particular purpose. We specifically
disclaim any liability with respect to this document and no
contractual obligations are formed either directly or indirectly
by this document. This document may not be reproduced or
transmitted in any form or by any means, electronic or mechanical,
for any purpose, without our prior written permission.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective owners.