

Unified Storage And Searching Of Structures And Relational Data In Oracle[®]: An Overview Of AUSPYX[®] 1.7

By *Webster Homer and Michael Mandl*
Tripos, Inc.

Table of Contents

EXECUTIVE SUMMARY	3
CHEMISTRY AND RELATIONAL DATABASES	3
THE AUSPYX DATA CARTRIDGE	4
INDEXING CHEMICAL STRUCTURES	5
SEARCH OPERATORS	6
<i>Substructure searching</i>	6
<i>Similarity searching</i>	7
<i>Exact match searching</i>	9
<i>Formula searching</i>	9
AUSPYX QUERY OPTIMIZATION	10
AUSPYX DATA TYPES	11
CONFORMATION STORAGE	12
REGISTRATION DATA	12
AUSPYX DATASETS	13
IMPORTING DATA INTO AUSPYX	14
EXPORTING DATA FROM AUSPYX	15
TRANSLATE_OPS UTILITIES	15
SLN_OPS UTILITIES	16
OPTISIM™ INTEGRATION	17
STORING AND SEARCHING 3RD PARTY FINGERPRINTS	18
FPRINT_OPS UTILITIES	18
DOMAIN INDEX MAINTENANCE TOOLS	18
SUMMARY	18

EXECUTIVE SUMMARY

Historically, relational databases such as Oracle have been unable to store, index, and search complex domain-specific data such as chemical structures. Recently, Oracle has provided mechanisms for extending the capabilities of the Oracle server. This document describes the AUSPYX 1.7 data cartridge, which is built on the Oracle extensibility architecture, and discusses how AUSPYX enables storage and searching of chemical structures and the relational data associated with them in a single Oracle database.

CHEMISTRY AND RELATIONAL DATABASES

Historically, the storage, indexing, and searching capabilities of relational databases such as Oracle have been applicable only to numbers, dates, and strings. These databases have not been able to store and search complex domain-specific data such as chemical structures. Even when chemical structure data was stored in a relational database the semantic meaning of the data resided not in the database, but in the applications accessing it.

This longstanding limitation of relational databases has meant that structures and their associated data have been stored in separate systems. Proprietary applications were needed to support chemical structures and the operations that chemists apply to them. Typically, these systems coupled a chemical database with specialized tools for creation of 2D and 3D search queries based on structure. Other data, including bioassay results and analytical determinations, were stored in a conventional relational database.

For IT administrators, support for chemical research has been synonymous with maintenance and synchronization of two separate data sources. For chemists and biologists, the lack of integration between structures and related data has meant complexity in constructing and executing queries.

With the release of Oracle8i™, Oracle has provided the Oracle extensibility architecture. This set of interfaces allows developers to extend Oracle's capabilities through the creation of "data cartridges". A data cartridge is a method of packaging functionality to make Oracle servers aware of data types and data operations unique to a given discipline or industry. Data cartridges can be installed into or de-installed from an Oracle server in modular fashion.

The Oracle server manages the basic relational tasks, such as associating molecular structures with property data, creating specialized views of the data, and data security. It also provides capabilities for basic data storage, query processing, data indexing, and optimization. A data cartridge defines new data types, their index structure, the operators that work on them, and methods of query optimization for the new data types. Through the extensibility architecture, the new capabilities are registered with the server, which then uses the data cartridge implementation instead of its own.

THE AUSPYX DATA CARTRIDGE

AUSPYX is a data cartridge that supports chemical structure storage, indexing, and searching, and by doing so, enables the creation of chemical relational databases. The result is that it is possible to search for chemical structures in Oracle databases using standard SQL queries in the same way searching is done for other Oracle data types.

Data cartridges are contained in a schema that defines its structure and content. The AUSPYX schema name is "C\$TRPSUDC1". This schema owns all of the AUSPYX code, but other users may execute it. This allows AUSPYX datasets to be created and maintained in many different schemas. Most of AUSPYX's PL/SQL code is marked with the clause "AUTHID CURRENT_USER" which assures that the code executes as the current user invoking the shared PL/SQL code. For example, when user 'NCI' invokes a search, the shared PL/SQL code stored in the C\$TRPSUDC1 schema executes as NCI. The C\$TRPSUDC1 schema does not contain end user data.

AUSPYX represents chemical structures as SYBYL Line Notation¹ (SLN) strings and stores them in VARCHAR2(4000) database columns after removing all coordinate and non-structure data from the SLNs. SLN is the structural language used by Tripos' SYBYL[®] and UNITY[®] products.

The extensibility architecture provides interfaces through which developers register a data cartridge's specialized capabilities with the database server. AUSPYX builds on the extensible operator interface to add operators that implement chemical structure searches. This interface allows Oracle to call user-defined functions that can interact with the Oracle optimizer and provide support for user-defined domain indices.

The ODCIIndex interface provides hooks to Oracle indexing code for index definition, maintenance, and scanning of the chemical data types, and ODCIStats provides hooks to Oracle's optimizer. Developers write functions and create an indextype within the data cartridge. An operator definition relates these functions to an interface, while the indextype associates the operator with an index definition and the ODCIIndex methods provided by the developer.

¹ S. Ash, M.A. Cline, R.W. Homer, T. Hurst, G.B. Smith. *SYBYL Line Notation (SLN): A Versatile Language for Chemical Structure Representation* J. Chem. Info. Comp. Sci. **37**:71-79 (1997). A full description of the language is contained in the SLN manual shipped with AUSPYX.

Indexing Chemical Structures

To facilitate searching of chemical structures, AUSPYX takes advantage of Oracle's extensible indexing API and adds support for indexing molecules. A limitation of Oracle's extensible indexing system, however, is that it does not support parallel searching.

AUSPYX defines the indextype, SLN_INDEX_TYPE, which supports the AUSPYX search operators. Owners of AUSPYX datasets should create SLN_INDEX_TYPE domain indices on the SLN column to improve search performance.

When a user issues the CREATE INDEX command, Oracle calls the AUSPYX create index method (ODCIIndexCreate), which creates the domain index tables and populates them.

Extensible Indexing: The Create Index command within Oracle

```
CREATE INDEX <index_name> ON <TABLE_NAME>(<sln_column>) INDEXTYPE IS  
C$TRPSUDC1.SLN_INDEX_TYPE [PARAMETERS('TABLESPACE=<table_space_name>  
DEFER_UPDATE=TRUE')]
```

Oracle does not support synonyms on indextype names so the AUSPYX schema name, C\$TRPSUDC1, must be specified with the indextype name. The PARAMETERS clause is optional. Users may specify the tablespace where the domain index tables will be created. If users do not specify a tablespace, Oracle will create the tables in the user's default tablespace. Deferred mode indexes are new in AUSPYX 1.7. Deferred mode provides the ability to offload much of the overhead that causes contention when a large number of compounds are registered while the dataset is actively searched. Structures are added to the index, but part of the index is not updated which allows for between 8 and 10 times faster database inserts, at a small cost to searching performance. Users then schedule an index update (using the ALTER INDEX statement) when the database is not heavily loaded.

Once an SLN_INDEX_TYPE index has been created, new compounds added to the database are automatically added to the index. Oracle will call the AUSPYX index insert method (ODCIIndexInsert) to add the compound to the index. If a user modifies a compound, Oracle will call the AUSPYX index update method (ODCIIndexUpdate) to modify the corresponding index data. Finally, if the user deletes the compound from an indexed table, Oracle will call the AUSPYX index delete method (ODCIIndexDelete) to remove the compound's index information.

AUSPYX implements the indextype "SLN_INDEX_TYPE" with the type "SLN_SRCH_METHODS". This type is the AUSPYX implementation of the Oracle ODCIIndex type. The SLN_SRCH_METHODS type along with the SRCH_OPS PL/SQL package implement domain indices and the functional implementation of the operators.

Search Operators

AUSPYX 1.7 provides five operators that perform structure searching. These operators can be used in SELECT lists and WHERE clauses or anywhere else SQL allows such constructs to appear.

AUSPYX creates its operators in the C\$TRPSUDC1 schema so that the schema name must be specified when referencing the search operator; for example, C\$TRPSUDC1.SEARCH2D.

ORACLE allows the creation of a synonym on operators. The most general way to do this is to create a PUBLIC synonym so that all users can access AUSPYX operators without specifying the AUSPYX cartridge schema. The examples in this document assume that such synonyms have been created.

Substructure searching

The AUSPYX *Search2D* operator performs substructure searching in Oracle. The syntax for *Search2D* is as follows:

```
Search2D(SLN_COLUMN_NAME, 'SLN_PATTERN' [,options] )
```

SLN_COLUMN_NAME must be the name of a database column that contains SLN strings or a literal SLN string to search.

SLN_PATTERN is an SLN string that serves as the pattern to match against the SLN column.

The supported options are:

CHARGE: Boolean, default is 'TRUE' and means that charge is to be considered when matching atoms.

STEREO_SEARCH: String, default is 'TRUE'. It means that stereochemistry is to be considered when matching atoms and bonds. "EXPLICIT" means: use explicit method. Query with relative chirality returns hits with relative chirality only. Query with unknown chirality returns hits with unknown chirality only. FALSE means: don't consider stereochemistry.

ISOTOPE: Boolean, default is 'TRUE'. It specifies that isotopes are to be considered when matching atoms.

CROSSFRAGMENT: Boolean, default is "TRUE". Permit multiple fragments in a query to match across more than a single target fragment. For example, the query O.O can match O[1]C=CC@1.CH3OH when set to TRUE, but it will not match if set to FALSE. (Default is TRUE.)

SCREENONLY: Boolean, default is "FALSE". Reports all structures that pass screening without actual 2D search, which is useful for determining the selectivity of a query against a dataset.

The *Search2D* operator returns a number "1" if the structure matches the pattern and "0" if it fails to match. It is possible that the operator can be passed a NULL SLN to search, in which case the operator will return NULL.

When a user submits a query using the *Search2D* operator on a column indexed with SLN_INDEX_TYPE, Oracle calls the AUSPYX index scan method (ODCIIndexStart). The scan method performs 2D screening based on the query and content of the AUSPYX index to eliminate rows from the queried table quickly. Once complete, a cursor is opened to select those candidates that have not been screened. Oracle will then repeatedly call the domain index fetch method (ODCIIndexFetch) to retrieve structures from this query while performing atom-by-atom searching to determine each match and return their ROWID to Oracle.

The following example query will return all the regIDs and their structures that contain an amide group.

```
SELECT
    r.regid, s.sln_string
FROM
    reg_tab r, primary_sln s
WHERE
    r.sln_id = s.sln_id AND
    Search2D(s.sln_string, 'NC=O') = 1;
```

The above SELECT statement illustrates how substructure search becomes just another part of a WHERE clause. AUSPYX will allow users to put more than one *Search2D* operator into an SQL statement. If a domain index of indextype SLN_INDEX_TYPE is present on the primary SLN table, Oracle can use it.

Similarity searching

AUSPYX provides two similarity search operators. The primary operator, *SIMILARITY*, implements the similarity search. It returns a number containing the percent similarity of the structure to the query. AUSPYX also supplies an ancillary operator, *SIMINDEX*, which reports the similarity value found by the primary *SIMILARITY* operator.

SIMILARITY

The syntax of the *SIMILARITY* operator is as follows:

```
SIMILARITY(SLN_COLUMN_NAME, SLN_PATTERN [,options])
```

As with the *Search2D* operator, the first argument can be the name of a database column containing SLN strings or a specific SLN string to search. The second argument is a string containing a valid SLN. The operator calculates the similarity of the structure to the pattern. If the column is indexed with an SLN_INDEX_TYPE, the index contains the

fingerprints for the SLNs stored in the SLN column being indexed. These fingerprints will be compared in the similarity calculation.²

Valid options are:

TYPE: AUSPYX 1.7 supports three types of similarity calculation: Tanimoto (the default), asymmetric, and cosine.

QUERY_WGT: used only by asymmetric similarity, this is a positive integer between 0 and 100 that is used to weight the query in the similarity calculation.

SIMINDEX

The *SIMINDEX* operator is an ancillary operator to the *SIMILARITY* operator. The *SIMINDEX* operator is useful for returning the similarity values in a select list for the similarity calculated in the WHERE clause. The syntax of the operator is as follows:

```
SIMINDEX(literal_number)
```

SIMINDEX returns a number. The single argument to *SIMINDEX* is a literal number used to relate *SIMINDEX* used in a statement to the primary operator, *SIMILARITY*, also used in the same statement. Therefore, when using *SIMINDEX*, the *SIMILARITY* operator allows an additional argument located after the options holding a number corresponding to the intended *SIMINDEX* operator clause. *SIMINDEX* will then return the actual similarity index of each matching row. When a user submits a query using the *SIMILARITY* operator, Oracle calls the AUSPYX index scan method (ODCIIndexStart). This method computes a fingerprint for the query. It then computes the maximum and minimum fingerprint cardinalities that compound fingerprints must have in order to match the query. The scan method creates a query of the domain index tables to return the matching compounds. Oracle then repeatedly calls the AUSPYX index fetch function (ODCIIndexFetch) that reads the query and evaluates the domain index fingerprints against the query fingerprint. If the similarity value falls within the specified range, the function returns the ROWID of the matching SLN to Oracle. Only the fingerprint of the largest structure in a mixture is stored and used by *SIMILARITY* search.

The following example selects the regID, SLN string, and similarity for all compounds that are greater than 85% similar to pyrrole using the Tanimoto similarity calculation:

```
SELECT
    r.regid, s.sln_string, SIMINDEX(1)
FROM
```

² A UNITY molecular fingerprint is a wide bit set where each bit indicates the absence or potential presence of a molecular feature. UNITY fingerprints are described in the UNITY manual by Tripos, Inc. See the AUSPYX documentation for more information about the similarity calculations employed.

```
Reg_tab r, primary_sltn s
WHERE
  r.sltn_id = s.sltn_id and
  SIMILARITY(s.sltn_string, 'N[1]H:CH:CH:CH:CH:@1', 1) > 85;
```

Exact match searching

EXACTMATCH finds all SLNs that are exact matches for each other. *EXACTMATCH* uses a hash code stored in the domain index to quickly find matching structures. *EXACTMATCH* has options similar to *Search2D*.

```
EXACTMATCH(SLN_COLUMN_NAME, SLN_PATTERN[, OPTIONS])
```

When a user submits an *EXACTMATCH* query on a column with a domain index, Oracle calls the AUSPYX index scan method (ODCIIndexStart). This method calculates a hash code for the query that selects only the structures with a matching hash code from the index. Oracle will then repeatedly call the AUSPYX index fetch method (ODCIIndexFetch) to fetch candidates from the query and do the exact match operation. It then returns the ROWID of each matching structure to Oracle.

Valid options are:

TAUTOMER: Search for tautomers of the query, default is False.

SUBSET: Find all compounds that contain the query fragment(s). Suitable for finding salts. The default is False.

CHARGE: Similar to the *Search2D* operator. Useful with subset option for finding salts. The default is True.

FREE_SALT: Removes counterions from the query prior to searching.

BOND_STEREO: Consider stereochemistry when matching bonds. Default=TRUE.

CHIRALITY: Consider chirality when matching atoms. Default=TRUE.

ISOTOPE: Consider isotopes when matching atoms. Default=TRUE.

NEUTRALIZE: Remove counter ions from the query SLN, then neutralize the remaining fragments. (Counterions are defined in the UDC_CONFIG_RULES_TBL "counter_ion" rule. This rule is loaded from the counterion.defs file.) Default is FALSE.

As with the *Search2D* operator, the *EXACTMATCH* operator returns 1 if a structure matches, and 0 if it does not.

Formula searching

The syntax for formula searching is:

```
FORMULA(SLN_COLUMN_NAME, FORMULA, OPTIONS )
```

FORMULA finds all structures that have molecular formulas matching the query formula. The first argument is the name of an SLN database column. The second argument is a molecular formula.

FORMULA has only one option:

SUBFORMULA: When set to true, this treats the molecular formula as a subformula. The default is false. Subformulas are partially specified molecular formulas. For example, a subformula query for “C5N” will return all compounds that have five carbons and one nitrogen. Subformula queries can make use of some non-elemental atom types such as Het or Hev. The query “C10Het3” will return all structures with ten carbons and three heteroatoms. Het is defined to be either oxygen, nitrogen, sulfur, or phosphorous.

AUSPYX Query Optimization

Oracle’s cost-based optimizer (CBO) generates an execution plan for a SQL statement that has the least cost among the possible execution plans. The cost of a plan is calculated from the cost of the access method on each table in the FROM clause, and the cost of the join order of the tables in the FROM clause. The optimizer chooses a plan by generating a set of join orders, computing the cost of each, and selecting the one with the lowest cost. For each table in the join order, the access method and the join method that has the least cost is chosen. Statistics collected from referenced tables and indices help the optimizer to estimate the cost and selectivity of query predicates.

The introduction of the extensible architecture for the support of user-defined access methods prevents the Oracle CBO from determining an accurate cost. This is because only developers working to extend Oracle understand the semantics of user-defined operators, functions, and types. Oracle codes are missing this key information. To address this issue, the Oracle extensibility framework allows application programmers to provide methods that communicate data to the CBO about domain indices, operators, and types.

AUSPYX provides an interface to Oracle’s CBO through the extensible optimizer or ODCIStats interface. This interface allows AUSPYX to provide functions that the CBO calls to determine the cost of AUSPYX operators used in SQL statements. The callbacks return information such as the selectivity of the operator, the CPU usage, and IO usage. Computation of these values requires that an index exists and that statistics are collected on some of these index tables. The CBO then determines the best join order of multiple predicate statements and hence the best execution plan for the statement based upon statistics.

A necessary part of this process is the collection of statistics upon which the ODCIStats interface implementation determines the cost of operators. The collection of statistics is controlled through the ANALYZE command. Users may collect statistics against specific AUSPYX indices by issuing ANALYZE TABLE or ANALYZE INDEX commands. However, since the efficient

operation of the operators depends upon statistics, it is recommended that underlying AUSPYX index tables have statistics present prior to collecting “domain” index statistics. For this reason, the STAT_OPS package exposes the COMPUTE_INDEX_STATS() function to mediate the collection of statistics in an optimal order.

It is difficult to create general rules upon which estimates of performance can reliably scale with dataset content and size, not to mention database configuration. This is because the AUSPYX operators depend upon intermediate SQL statements in their operation. Consequently, their operation varies with the content of data and system performance. Therefore, collected statistics are based upon the actual execution of various SQL statements making use of AUSPYX operators, while measuring specific portions of the underlying codes. Investigation into the operation of AUSPYX operators has determined the discrete contributors to total cost and how these costs can be combined to yield a good estimate of actual total cost.³

AUSPYX Data Types

SLN_ID

This column is the unique identifier for the SLN record. It is assigned when the SLN is inserted into the database. AUSPYX uses an Oracle Sequence to generate SLN_IDs.

SLN_STRING

The SLN_STRING column contains the SLN representation of a molecule. Oracle 8i limits the varchar2 data type to 4000 bytes, so AUSPYX is unable to store SLNs larger than 4000 characters. Coordinates, name, regID, and other nonstructural information are stripped from the SLN prior to storing it.

During the process of loading, SLNs are normalized and uniqued. This process normalizes the aromatic systems, and standardizes idiomatic structures such as nitro groups into a normal form. Subsequently, uniquing of SLNs takes into account all atom and bond types and attributes. Uniqued structures simplify duplication checking and retrieval from the database. Mixtures are stored together in the same SLN.

CRC

The CRC is a hash number calculated from the unique SLN. It is used as an index for retrieving the SLN. While the CRC is not unique there are typically very few collisions, and when collisions occur the SLNs that the CRC is derived from are very different from each other. Note that AUSPYX provides one of two selectable duplicate checking methods: Regular and Tautomer. This CRC corresponds to the chosen duplicate checking method.

³ For more information, see the AUSPYX manual.

COORD2D

The COORD2D is a BLOB containing the 2D coordinates for the atoms in the SLN. The BLOB contains an array of single precision IEEE floating point numbers, with two values per atom in the SLN. The coordinates are written in the same order as the corresponding atoms in the SLN. The floating point numbers representing the 2D coordinates are written into the BLOB in network byte order.

Return all of the SLNs with their 2D coordinates that are 85% similar to toluene.

```
SELECT s.get_sln_2d(), SIMINDEX(1)
FROM primary_sln s
WHERE SIMILARITY(s.sln_string,
    'CH3C[1]:CH:CH:CH:CH:CH:@1',1) >= 85;
```

SLN_ATTR

The SLN_ATTR is a CLOB containing non structure information from the SLN. Typically this will be used to store SLN CT attributes that are not searchable. The SLN_ATTR can store information translated from MDL SGROUP data to support translation to and from MDL MOL files while retaining MUL and super atom SGROUP information.

Conformation Storage

AUSPYX allows multiple conformations for each molecule stored. The default AUSPYX schema stores 3D conformation data in its own table with the following columns:

COORD_ID

COORD_ID is a unique identifier for all coordinate sets stored in the dataset. It is derived from an Oracle sequence.

SLN_ID

SLN_ID is a foreign key to the SLN table that relates the coordinate set back to the SLN associated with these coordinates.

LABEL

The label column is a text field used to assign a label to the coordinate set.

COORD3D

The COORD3D BLOB contains a conformation of the molecule. The conformation is stored as an array of single precision IEEE floating point numbers, with three values per atom in the SLN string. The values are stored in the BLOB in network byte order.

Registration Data

Registration data is stored in the REG_TAB table. The REG_TAB table is used to associate the structure data in the SLN and coordinate tables

with the primary registration identifiers. The following columns are contained in the REG_TAB table:

REGID

The REGID is the primary key of the registration table. It is an alphanumeric identifier of up to 256 bytes. Typically, the REGID is an identifier that the organization uses to identify database structures. This column is the link between AUSPYX datasets to the outside world or to relate structures to other columns held in the Oracle database. This column must be used in a join clause when returning structure and property data from SQL. For example:

```
SELECT
  p.ClogP, s.sln_string, r.regid, r.name
FROM
  Mol_prop p, primary_sln s, reg_tab r
WHERE
  r.regid = p.regid
  AND r.sln_id = s.sln_id
  AND Search2d( s.sln_string, 'NC=O' ) = 1;
```

SLN_ID

The SLN_ID is a foreign key to the SLN table.

NAME

This field holds the compound name.

DATE_STAMP

The date stamp is the date/time stamp for the registration record. This field contains the date and time when the compound was last inserted or updated.

AUSPYX Datasets

An AUSPYX dataset is a collection of tables created for the purpose of holding end user chemical structure data upon which the shared AUSPYX code operates. AUSPYX provides the ability to take chemical structures in common molecule formats such as SLN, MOL file and SMILES and place them into the dataset where they can be searched and related to other data stored in Oracle.

Customers create Oracle schemas through the use of SQL scripts contained in the AUSPYX common area (/TRPS/UDC1/sql). These scripts reference the shared AUSPYX code as well as standard SQL to create the schema and the objects required to represent an AUSPYX dataset. The username of the schema holding an AUSPYX dataset is termed the registrar, who has full access to data stored here. Other users may be granted access to the data owned by the registrar as desired. AUSPYX also provides the ability to convert/extend an existing Oracle schema to an AUSPYX registrar schema.

There may be at most one AUSPYX registrar schema per Oracle schema and so all the chemical structures stored in this schema are related. Storage of unrelated data requires an additional AUSPYX registrar schema. For example, NCI and MAYBRIDGE may be held in separate AUSPYX registrar schemas. Each AUSPYX registrar schema contains a registration table, and one or more AUSPYX datasets. The registration table is named REG_TAB. An AUSPYX structure dataset consists of two tables: an SLN table and its corresponding 3D coordinate table. The SLN table is of type SLN_T and is named PRIMARY_SLN by default but can be named differently if desired. The coordinate table's name is derived from its corresponding SLN table. The default name is PRIMARY_SLN_COORD.

The registration table is provided as the means through which chemical structures stored in an AUSPYX dataset relate to the outside world or to other tables in Oracle. REG_TAB contains a REGID column and SLN_ID column to allow users to join structure data to other related tables either directly or through some mapping such as a view or SQL operator, as is commonly done in Oracle databases.

In order to improve the AUSPYX product, future versions may eliminate the use of type tables completely. In such a change, the structure of the tables will remain the same, containing columns of the same type. Therefore the new tables can be used equivalently to support the SQL statements described in this paper. The one limitation is in use of methods directly on tables, such as:

```
SELECT s.get_atom_count() from primary_sln s where sln_id
between 1 and 50;
```

Note that those methods are equivalently available through the SLN_OPS and other AUSPYX packages. The previous example can be equivalently coded as:

```
SELECT SLN_OPS.get_atom_count() from primary_sln where
sln_id between 1 and 50;
```

IMPORTING DATA INTO AUSPYX

AUSPYX includes a PL/SQL package, UDC_IMPORT, that provides a programmatic interface allowing users to load their data into AUSPYX datasets. These routines may be called from any application that can execute PL/SQL blocks.

In addition, AUSPYX also provides a command line client application named "udcimport" that allows users to import data from common molecule file formats into an AUSPYX dataset. This application provides several options to control the import operation.⁴

⁴ See the AUSPYX documentation for additional details.

The SLN language can encode 2D structures and 3D conformations as well as registration information. This additional information will make SLNs quite long. UDC_IMPORT can handle SLNs as either VARCHAR2 or CLOB data. The package provides an interface to parse the SLN, and import it into an AUSPYX dataset. Users can select how duplicate data is handled, whether to skip the molecule or replace it in the dataset. The function UDC_IMPORT.PARSE parses the SLN into a context handle. This handle is then passed to the other UDC_IMPORT routines that ascertain if the structure is already in the database.

The AUSPYX TRANSLATE_OPS package contains procedures and functions that can parse VARCHAR2 strings containing MEDCHEM SMILES or CLOBs containing MOL Files into SLNs. The resulting SLN can then be passed to UDC_IMPORT routines for import into AUSPYX.

EXPORTING DATA FROM AUSPYX

AUSPYX provides a command line client application named "udcsearch" that allows users to export data from an AUSPYX dataset into SLN hitlists similar to those generated by SYBYL and UNITY. This application allows the user to report structures based on various search criteria including all structures, substructure search, similarity search, 'invalid' structures, structures without 3D coordinates, and others.

Users may also use the Oracle exp and imp utilities to export AUSPYX datasets to other Oracle schemas. Note that exp/imp will not preserve AUSPYX indices and those indices must be dropped prior to using exp. Once the new dataset is imported via imp, the domain index must be created via the "CREATE INDEX...indextype is c\$trpsudc1.SLN_INDEX_TYPE" command. Future versions of AUSPYX will provide a mechanism to facilitate efficient export and import of AUSPYX indices to eliminate the need to recreate the AUSPYX index.

TRANSLATE_OPS UTILITIES

The TRANSLATE_OPS package provides functions to translate to and from SMILES strings as well as to parse CLOBs containing MDL molecule data or SYBYL MOL2 molecules into SLNs.

FROM_MDL_MOL

Function to parse a CLOB containing an MDL MOL file and returns an SLN. This function is overloaded to support SLNs as VARCHAR2 or as CLOB data.

TO_MDL_MOL

Procedure to create a CLOB containing an MDL MOL file from an SLN

FROM_SMILES

Function to parse a VARCHAR2 string containing a MEDCHEM SMILES string and returns an SLN.

TO_SMILES

Function to generate a VARCHAR2 containing an MEDCHEM SMILES string from an SLN.

FROM_SYB_MOL2

Functions to generate a VARCHAR2 or CLOB containing an SLN from a CLOB containing a SYBYL MOL2 file.

TO_SYB_MOL2

Functions to generate a CLOB containing a SYBYL MOL2 file from an SLN CLOB, or an SLN and 3D coordinate data.

SLN_OPS UTILITIES

AUSPYX provides a set of utility functions for handling SLNs. The SLN_OPS functions are suitable for creating function-based indexes on SLN database columns. The SLN_OPS package provides all the functionality to allow users to construct Lipinski “Rule of Five”⁵ structure filters. The following is a brief description of some of the SLN_OPS functions and procedures.

GET_2D_SLN_STRING

Function to return the SLN with 2D coordinates. If the function is passed a BLOB with 2D coordinates these will be used, otherwise the function will generate a reasonable depiction of the molecule.

GET_ATOM_COUNT

Function returns the number of atoms in the SLN.

GET_BOND_COUNT

Function to return the number of bonds in the SLN.

GET_ROT_BOND_COUNT

Function to return the number of rotatable bonds in the SLN.

GET_RING_COUNT

Function to return the number of rings in the SLN.

GET_FRAGMENT_COUNT

Function to return the number of disconnected molecules in the SLN.

GET_FRAGMENT_LIST

Function to return a collection containing the individual disconnected fragments as SLNs.

GET_ISOTOPIC_MASS

Function to compute the isotopic mass of the SLN.

⁶ Lipinski CA, Lombardo F, Dominy BW, Feeney PJ. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced Drug Delivery Reviews*. 46(1-3): 3-26, Mar 2001

GET_LIPINSKI_DONOR_COUNT

Function to return the number of nitrogen and oxygen atoms that can donate a hydrogen for hydrogen bonding.

GET_LIPINSKI_ACCEPTOR_COUNT

Function to return the number of nitrogen and oxygen atoms that can accept a hydrogen bond.

GET_MLOGP

Function to compute the Moriguici LOGP (MLOGP) of a molecule. The Moriguici LogP⁶ is an analog of CLOGP.

GET_MOL_WEIGHT

Function to return the molecular weight of the SLN.

MOL_FORMULA

Function to return the molecular formula of the SLN. The molecular formula is expressed in Aldrich Notation.

IS_VALID_SLN

Function returns 1 if the SLN is valid. The function checks that atom valences are reasonable for the atom and that the molecule does not match rules in the invalid configuration rule in the schema's UDC_CONFIG_RULES_TBL.

MOL_COMPOSITION

Returns a string containing the percent composition of the molecule..

STRIP_HYDROGEN

Removes Hydrogen from input SLN. A Keepmetal flag determines if hydrogens should be removed from metals.

OPTISIM™ INTEGRATION

AUSPYX provides the ability to select diverse yet representative subsets of the database using Tripos Optimis™ technology. This is the same technology available in Tripos' Benchware® HTS DataMiner product. AUSPYX integrates Optimis with its FPRINT_OPS functions to support the use of Optimis with third party fingerprint data.

In addition to diversity selection, AUSPYX supports compound clustering based on the Optimis algorithm. To facilitate this, the OPTISIM_SELECT procedures of the UDC_OPTISIM package have been modified to take additional, optional parameters specifying:

- the table into which the cluster information is to be written,
- a cluster radius used to determine cluster membership, and

⁷ Moriguchi, I.; Hirono, S.; Liu, Q.; Nakagome, I.; Matsushita, Y. "Simple Method of Calculating Octanol/Water Partition Coefficient" Chem. Pharm. Bull., 40, 127-139 (1992).

- an argument indicating whether or not distances of each clustered compound to its signpost (cluster ID) are to be reported.

In addition, two new procedures have been added to UDC_OPTISIM to partition an AUSPYX dataset using previously obtained signposts. The two procedures are named OPTISIM_PARTITION, one for AUSPYX (UNITY) fingerprints and one for application-specified fingerprints.

STORING AND SEARCHING 3RD PARTY FINGERPRINTS

The FPRINT_INDEX_TYPE allows AUSPYX to index and search 3rd party fingerprints. Built on top of FPRINT_OPS, this indextype provides a new set of operators: Tanimoto, Cosine, Tversky, Hamming, Asymmetric, and Dice methods of similarity. This fingerprint index type supports both traditional bitstring fingerprints stored as RAW or BLOB columns as well as SciTegic® fingerprints when loaded into the new TY_FPRINT_SPARSE Oracle type.

Standard UNITY and Atom Pair fingerprints can be created using FPRINT_OPS and indexed with the FPRINT_INDEX_TYPE.

FPRINT_OPS UTILITIES

The FPRINT_OPS package provides a suite of routines for handling fingerprint data including fingerprints data from third party developers. The suite includes functions to compute a number of similarity methods: Tanimoto, Cosine, Tversky, Hamming, Asymmetric, Modified Tanimoto and Dice. Users can develop their own metrics using FPRINT_OPS functions, and can encode and decode fingerprint data from several ASCII encoding methods. The package supports the storage of fingerprints in both RAW and BLOB column types.

DOMAIN INDEX MAINTENANCE TOOLS

AUSPYX 1.7 includes a package of tools for handling the SLN_INDEX_TYPE tables and data structures. These tools allow administrators the ability to backup and restore domain indexes, recover corrupted indexes, move an index to a different tablespace, and compress an index thereby recovering space from deleted records.

SUMMARY

Implemented on Oracle's extensibility architecture, AUSPYX enables storage and searching of chemical structure within Oracle. By creating a chemical relational database, AUSPYX allows users to build data repositories that unite chemical structure data with biological and analytical data. AUSPYX removes the need to maintain separate chemical databases, thereby reducing the associated administrative burdens and simplifying access to the full range of information necessary for making research decisions.

For more information

Please contact your Tripos Discovery Informatics consultant, call 1-800-323-2960 in the US, send email to contact_us@tripos.com, or visit our web site www.tripos.com.

USA
Tripos, Inc.
1699 South Hanley Road
St. Louis, MO 63144
1-314-647-1099

FRANCE
+33 1 69 59 29 49

GERMANY
+49 89 45 10 300

AUSTRALIA
+ 61 (7) 5439 9775

UNITED KINGDOM
+44 1908 650000

CANADA
+1 450 4334500

Tripos combines leading-edge technology and innovative science to deliver consistently superior chemistry-research products and services for the biotechnology, pharmaceutical and other life science industries. Within Tripos' Discovery Informatics (DI) business, the company provides software products and consulting services to develop, manage, analyze and share critical drug discovery information. Within Tripos' Discovery Research (DR) business, Tripos' medicinal chemists and research scientists partner directly with clients in their research initiatives, leveraging state-of-the-art information technologies and research facilities.

While the information presented in this white paper is believed to be current and accurate, it is presented as-is without guarantees or warranties of any kind. Tripos assumes no obligation to update any information in this document. Tripos makes AUSPYX available subject to the terms of the Tripos Software License Agreement, and nothing in this white paper should be construed as a warranty concerning the performance of the product

AUSPYX, SYBYL, UNITY, and the Tripos logo are trademarks or registered trademarks of Tripos, Inc. All other trademarks are the property of their respective owners.

©2006 Tripos, Inc. All rights reserved.

Printed in USA

