



**Dr. Warren Lyford DeLano**  
**June 21, 1972 - November 3, 2009**

On Tuesday, Nov 3, 2009 Warren DeLano passed away suddenly at his home. He was 37 years old.

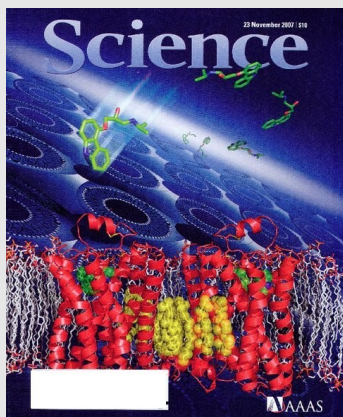
Through PyMol and Open Source software, Warren DeLano exhibited the genius and generosity of science at its best. Warren was committed to the development of Open Source programs and how they would benefit humanity by allowing science to flourish in a collaborative environment.

Warren DeLano was born in Philadelphia, raised in Palo Alto, and educated at Yale University. At Yale, he joined Axel Brunger's lab, where he made critical contributions to the computational tools and methods that made "CNS" a universal computational crystallography program.

After Yale, Warren received his doctorate at the University of California, San Francisco under the mentorship of Jim Wells, PhD. He then became a founding scientist at Sunesis Pharmaceuticals, creator of PYMOL, and founder of DeLano Scientific.

Throughout his life and career, Warren made fundamental and highly valued contributions to science. His Open Source PyMOL software is widely used throughout the world, and nearly all publications that display macromolecular structures use PyMOL.

He is survived by his wife Beth Pehrson, mother Margaret DeLano, father James DeLano, Jr., step-mother Cathy Groves DeLano, step-father Tom Snouse, sister Jennifer DeLano and brother Brendan DeLano, as well as three aunts and eight cousins. Memorial service is pending.



# 5. German Conference on Cheminformatics Free Software Session

- OrChem: An open source chemistry search engine for Oracle
- Bingo: A data cartridge for Oracle database for fast, scalable, and efficient storage and searching solution for chemical information.
- Dingo: A IUPAC-compliant cross-platform open-source library for molecule and reaction 2D structural formula rendering

# OrChem

An open source chemistry search engine for Oracle

Sept 2009

Mark Rijnbeek and Christoph Steinbeck  
Cheminformatics and Metabolism Team , EBI



# OrChem overview

- OrChem is an open source chemistry search engine for Oracle (11g)
- Facilitates substructure and similarity searching on compound data
- Built on top of the Chemistry Development Kit (CDK)
- Free, open source product
- Download & (installation) documentation : <http://orchem.sourceforge.net>



OrChem sits under the Sourceforge 'ChemiSQL' umbrella, which is to result in a common database API (eventually)

- Mychem for MySql
- Pgchem for Postgres
- Orchem for Oracle



<http://sourceforge.net/projects/chemdb/>

Software

Open Access

**Journal of  
Cheminformatics**  
Volume 1

# OrChem - An open source chemistry search engine for Oracle

Mark Rijnbeek  and Christoph Steinbeck *Journal of Cheminformatics* 2009, **1**:17 doi:10.1186/1758-2946-1-17

Published: 22 October 2009

## Abstract (provisional)

### Background

Registration, indexing and searching of chemical structures in relational databases is one of the core areas of cheminformatics. However, little detail has been published on the inner workings of search engines and their development has been mostly closed-source. We decided to develop an open source chemistry extension for Oracle, the de facto database platform in the commercial world.

### Results

Here we present OrChem, an extension for the Oracle 11G database that adds registration and indexing of chemical structures to support fast substructure and similarity searching. The cheminformatics functionality is provided by the Chemistry Development Kit. OrChem provides similarity searching with response times in the order of seconds for databases with millions of compounds, depending on a given similarity cut-off. For substructure searching, it can make use of multiple processor cores on today's powerful database servers to provide fast response times in equally large data sets.


### Availability

OrChem is free software and can be redistributed and/or modified under the terms of the GNU Lesser General Public License as published by the Free Software Foundation. All software is available via <http://orchem.sourceforge.net>.

**Viewing options:**

- **Abstract**
- **PDF (629KB)**

**Associated material:**

- Readers' comments 

**Related literature:**

- Other articles by authors
  - ⊕ on Google Scholar
  - ⊕ on PubMed
- Related articles/pages
  - on Google
  - on Google Scholar

**Tools:**

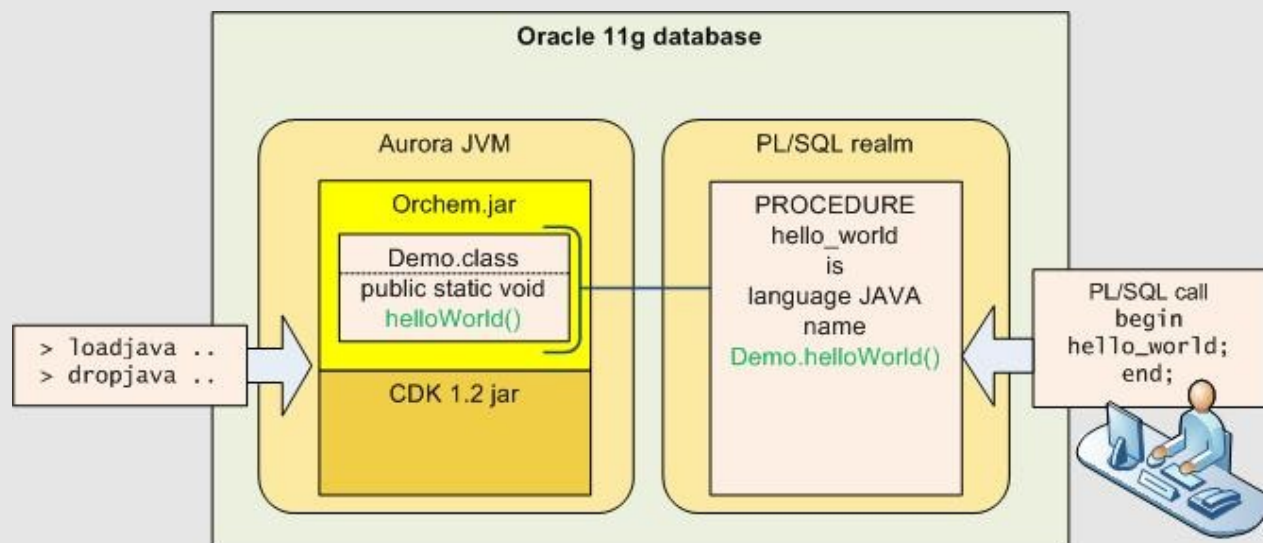
- Download citation(s)
- Email to a friend
- Order reprints
- Post a comment

 **Nominate for award****Post to:**

-  Citeulike
-  Connotea
-  Del.icio.us
-  Facebook
-  Twitter

# OrChem concept

- OrChem is not a *cartridge* but uses standard features.  
note: for Oracle various commercial chemistry data cartridges exist (=external libraries)
- OrChem uses the Oracle *internal JVM* Aurora (Java 1.5) that allows Java to be loaded and executed inside the database.
- JVM performance improved (JIT) with release 11g, but still not ultra fast.
- All Java has to be invoked through native *PL/SQL* wrappers.



## Some query examples

Example: a substructure search for a SMILES pattern, cut-off 250 results:

```
select id from table
  (orchem_subsearch.search('C:C:C:O', 'SMILES', 250, 'Y') )
```

Example: a substructure search for a SMILES pattern, 40% similarity cut off:

```
select * from table
  (orchem_simsearch.search('CCCNC', 'SMILES', 0.4) )
```

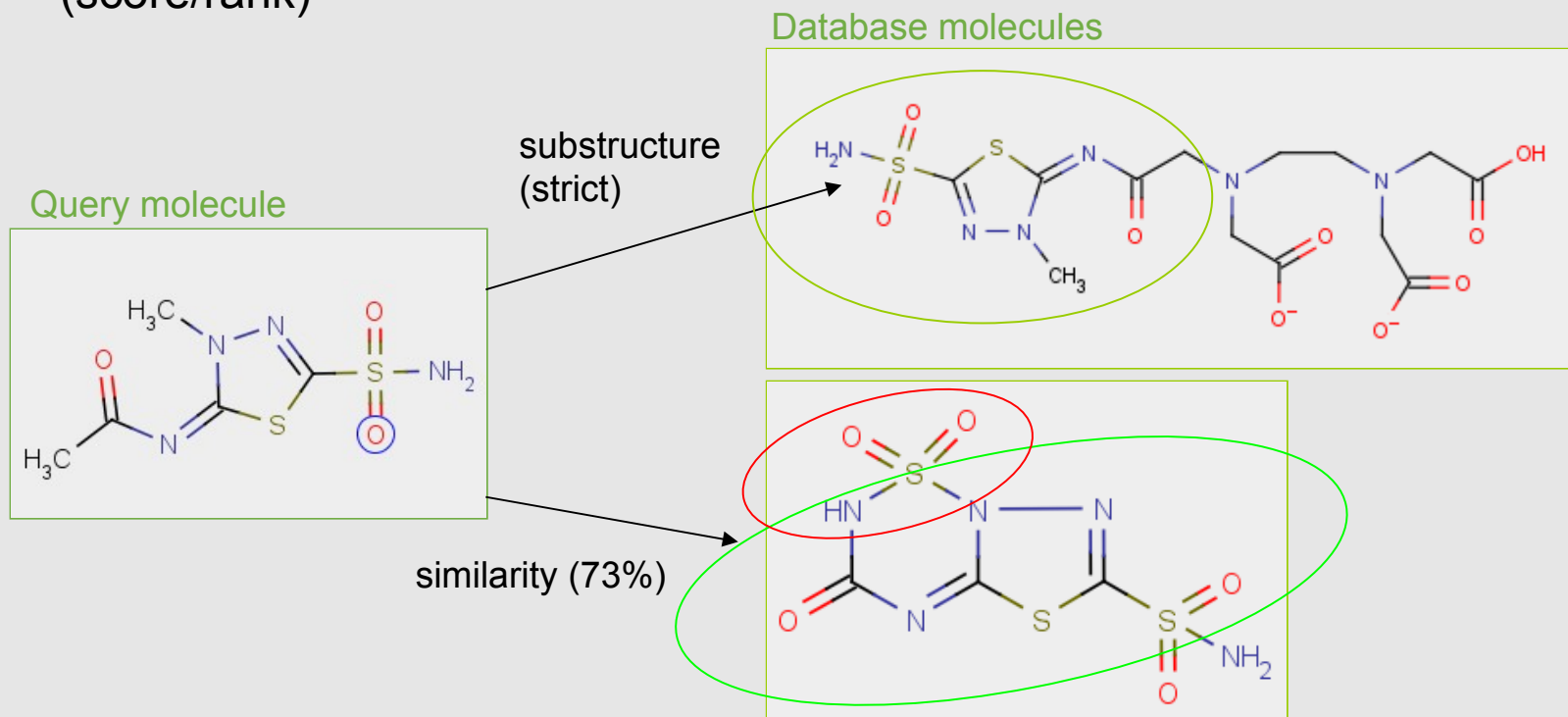
Example: produce a Mol-file descriptor from a SMILES descriptor:

```
select orchem_convert.smilestomolfile('ClCCOP(=O)(OCCCl)OCCCl')
  from dual
```

# Using OrChem to search compounds

OrChem supports two types of compound (graph) searching:

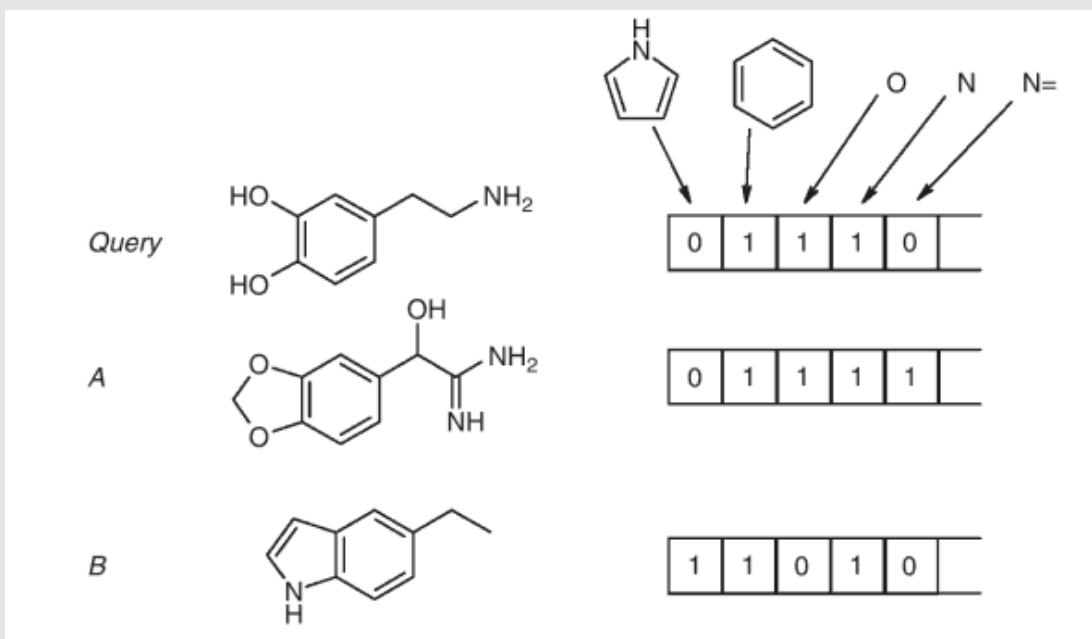
- a *substructure search* identifies all the molecules that contain a specified substructure
- a *similarity search* identifies all molecules similar to a specified structure (score/rank)

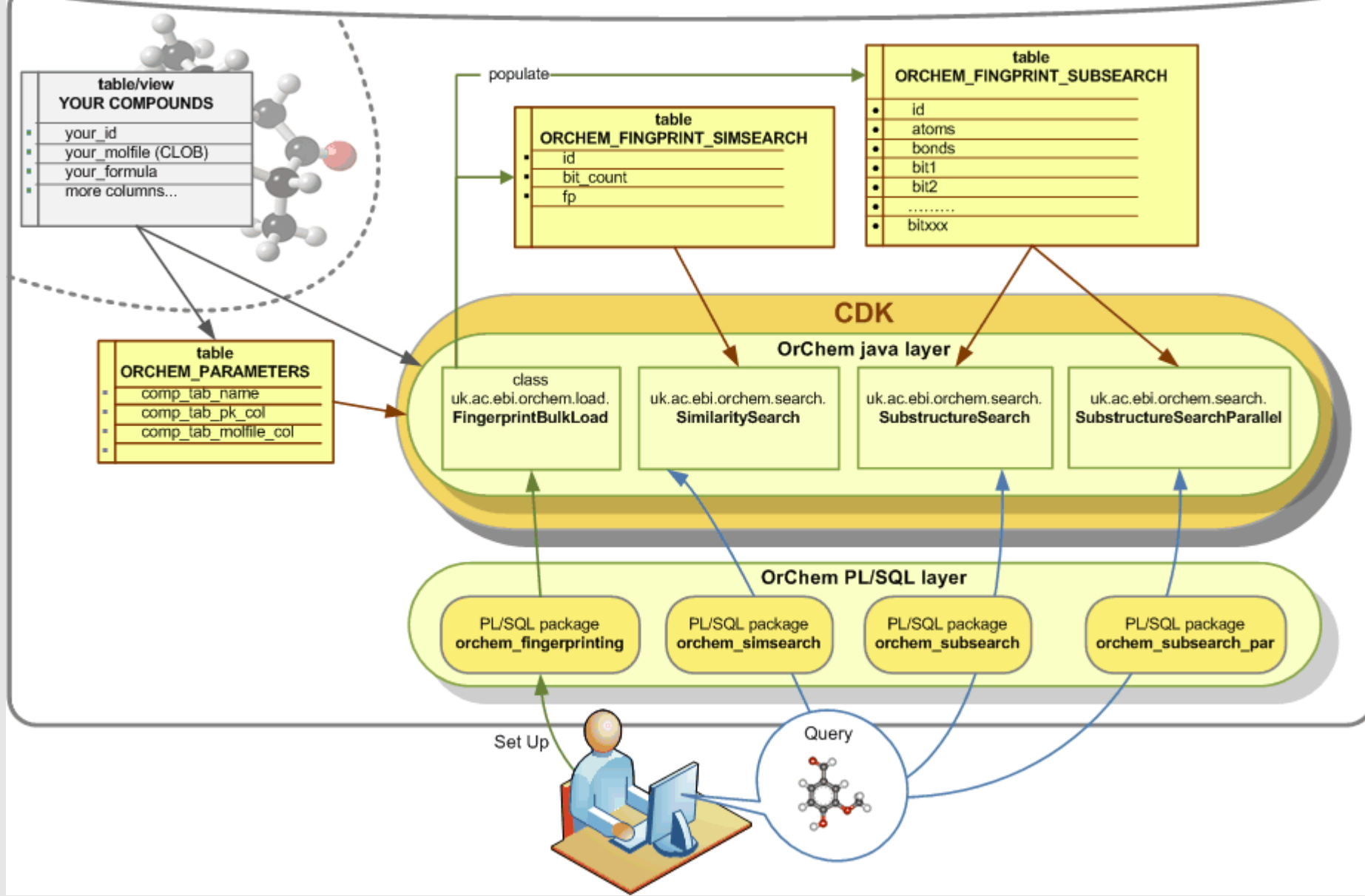




# Fingerprints are used for compound searching

Fingerprints are bit sets that consist of a sequence of “0”s and “1”s: a one indicates the presence of a particular structural feature, a zero indicates its absence.



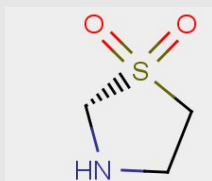


# Substructure searching is a two-step process

- Step one: “pre-screening” narrows down the set of possible candidates. The fingerprint is used to filter away impossible options: if a query is to be a *substructure* of some database molecule, then all the bits set to 1 in the query must be set to 1 in the database molecule
- Step two: the pre-screened candidates still needs to be verified to contain the query using a computationally more expensive *graph isomorphism* algorithm
- The trick is to let the pre-screening work fast, be precise, and result in a high percentage of positive isomorphic verifications.

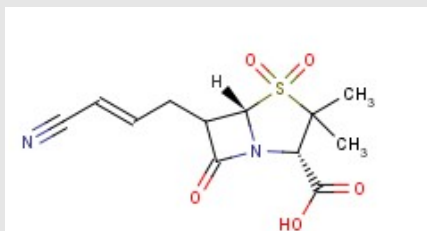
# Substructure search pre-screen example

Suppose the fifth bit in a fingerprint is set whenever O=S=O occurs in a compound. The query compound is this one:

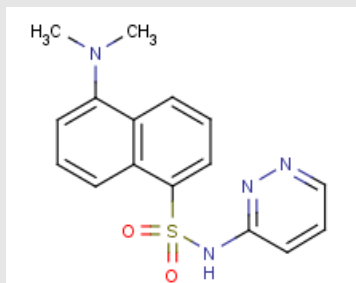


0	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

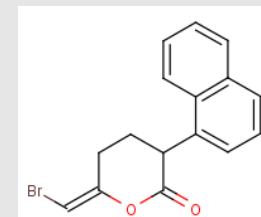
The pre-screening will filter away the third candidate because the fifth bit is zero.



0	0	0	0	1	0	1	1
---	---	---	---	---	---	---	---



0	0	0	0	1	1	1	1
---	---	---	---	---	---	---	---



0	0	1	0	0	0	0	0
---	---	---	---	---	---	---	---

# OrChem substructure search example

Example: find (Smiles based) compounds that have a carbon triangle for a *substructure*, cap to 100 results.

```
q4> select id from table (orchem_subsearch.search('C1CC1', 'SMILES', 100))
```

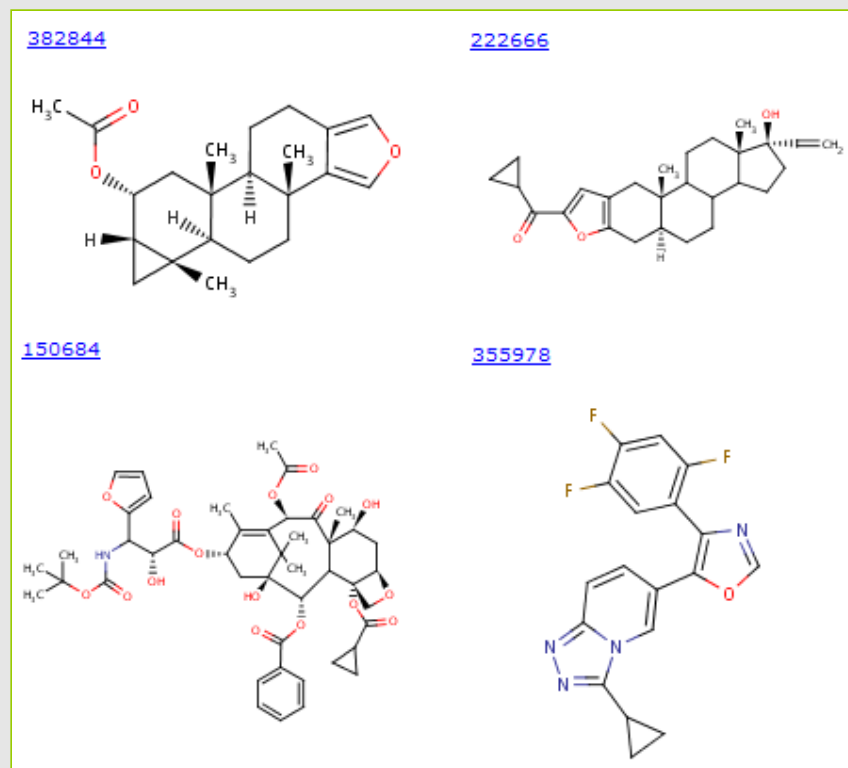
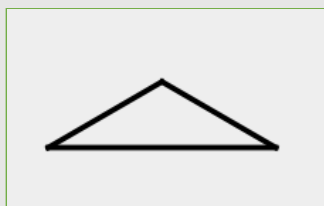
ID

-----  
382844

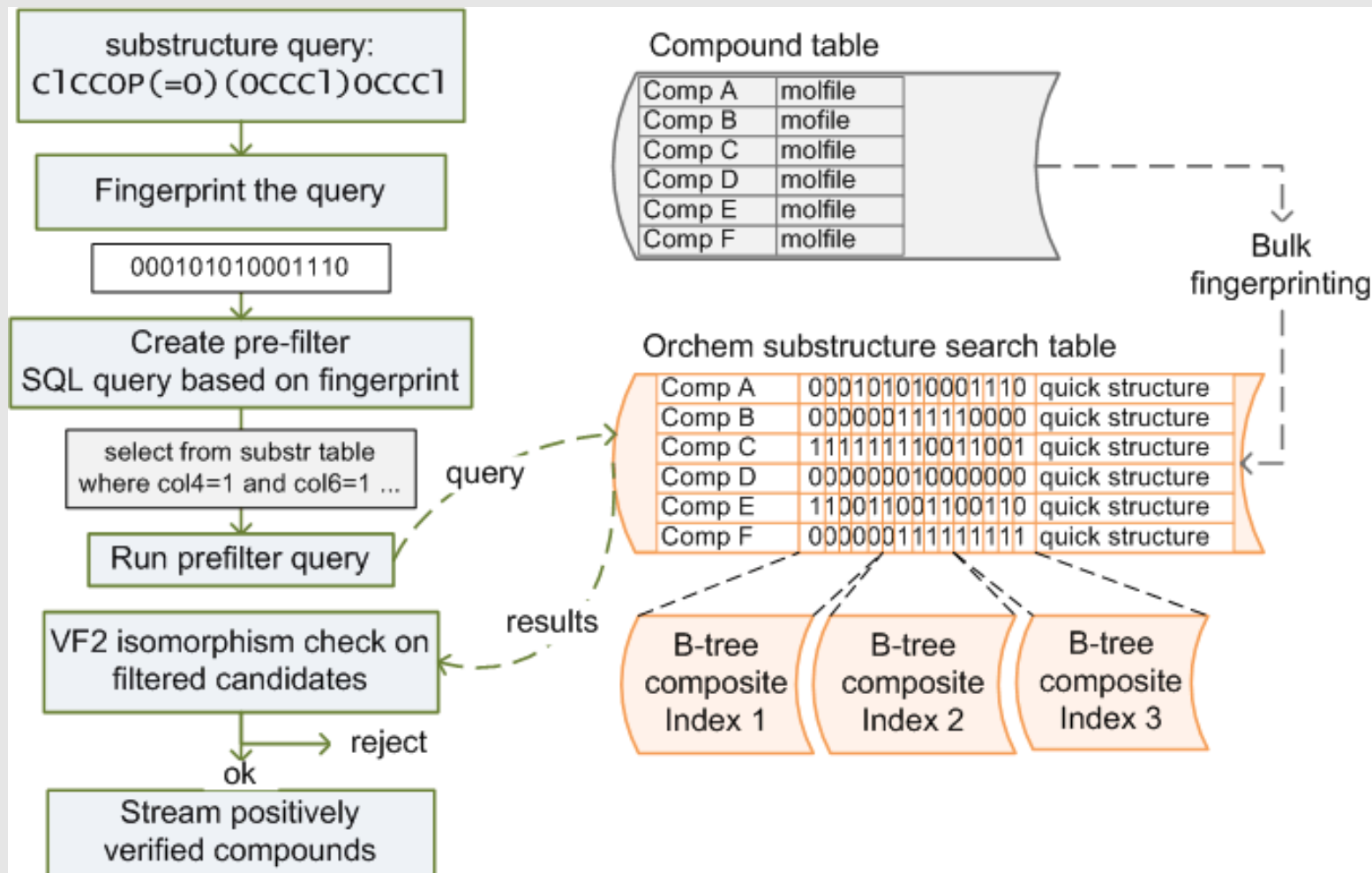
222666

150684

...



# Substructure search implementation



# Substructure search optimization

- B-tree indexes on fingerprint table. This enables faster pre-filtering; the Oracle Optimizer figures out the appropriate index

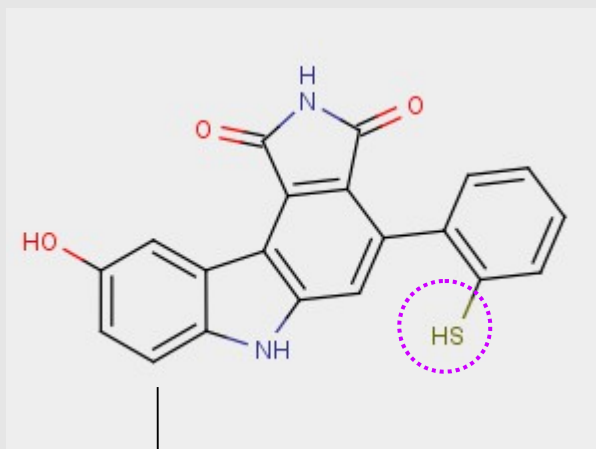
```
q5> select id from orchem_fingerprint_subsearch where bit311='1';
```

## Execution Plan

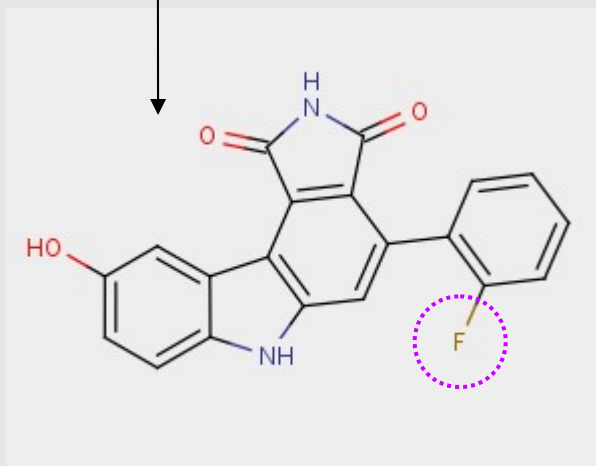
Id	Operation	Name
0	SELECT STATEMENT	
1	TABLE ACCESS BY INDEX ROWID	ORCHEM_FINGERPRINT_SUBSEARCH
* 2	INDEX SKIP SCAN	ORCHEM_BTREE10

- Parallel substructure search option (pros/cons: see paper)
- Fast isomorphism check with “VF2”, a recursive backtracking graph comparison algorithm.
- Added VF2 optimization: “sorting” molecules (atom containers) based on uniqueness of elements and bond types. Standard algorithm is not aware of different node/edge types.

# Sorting atom containers before VF2 isomorphism



Substructure? No.



## Unsorted - 107 steps:

- 1: O-O
- 2: .C-C
- 3: ..C-C
- 4: ...C-C
- 5: ....C-C
- 6: .....N-N
- 7: .....C-C
- 8: .....C-C
- 9: .....C-C
- 10: .....C-C
- 11: .....C-C
- 12: .....C-C
- 13: .....C-C
- 14: .....O-O
- 15: .....N-N
- 16: .....C-C
- 17: .....O-O
- 18: .....C-C
- 19: .....C-C
- 20: .....C-C
- 21: .....C-C
- 22: .....S-F
- 23: .....S-C
- 24: .....S-C
- 25: .....C-C
- 26: .....C-C
- 27: .....C-C
- 28: .....O-C
- .....
- ..
- 105: O-C
- 106: O-C
- 107: O-C
- no match

## Sorted - 26 steps:

- 1: S-F
- 2: S-N
- 3: S-N
- 4: S-O
- 5: S-O
- 6: S-O
- 7: S-C
- 8: S-C
- 9: S-C
- 10: S-C
- 11: S-C
- 12: S-C
- 13: S-C
- 14: S-C
- 15: S-C
- 16: S-C
- 17: S-C
- 18: S-C
- 19: S-C
- 20: S-C
- 21: S-C
- 22: S-C
- 23: S-C
- 24: S-C
- 25: S-C
- 26: S-C
- no match



# Similarity search and fingerprinting

For *similarity searching*, the same fingerprint as for substructure searching can be used. One step only – no isomorphism required.

- if a query compound Q has fingerprint bitset Bq
- and a target compound T has fingerprint bitset Bt
- Then we can calculate Tanimoto similarity between Q and T being

$$S_{QT} = c / (a + b - c) \quad (0..1 \text{ score})$$

**a**=bits set to 1 in Bq, **b**=bits set to 1 in Bt, **c**=1 bits common to Bq and Bt

- Example

```
Bq  1100111000    a=5
Bt  0101111011    b=7
-----
BitAnd  0100111000    c=4  Similarity= 4 / (5+7-4) = 0.5
```

# Similarity search implementation

- Similarity searching is done by a Java stored procedure
- Algorithm is described in a 2007 paper by Swamidass and Baldi, it allows a focused search on most likely candidates
- Orchem table to support similarity algorithm:

```
$ desc orchem_fingerprint_simsearch
ID          NOT NULL VARCHAR2(80)
BIT_COUNT   NOT NULL NUMBER(4)
FP          NOT NULL RAW(100)
```

- Column “BIT\_COUNT” is bitmap indexed
- Start: use BIT\_COUNT to inspect compounds for which the number of bits set to one is the same as that of query molecule.
- Then, inspect compounds with a bit count closest to that of the query until done, that is until the minimum similarity has been reached or until the result set size satisfies the user.

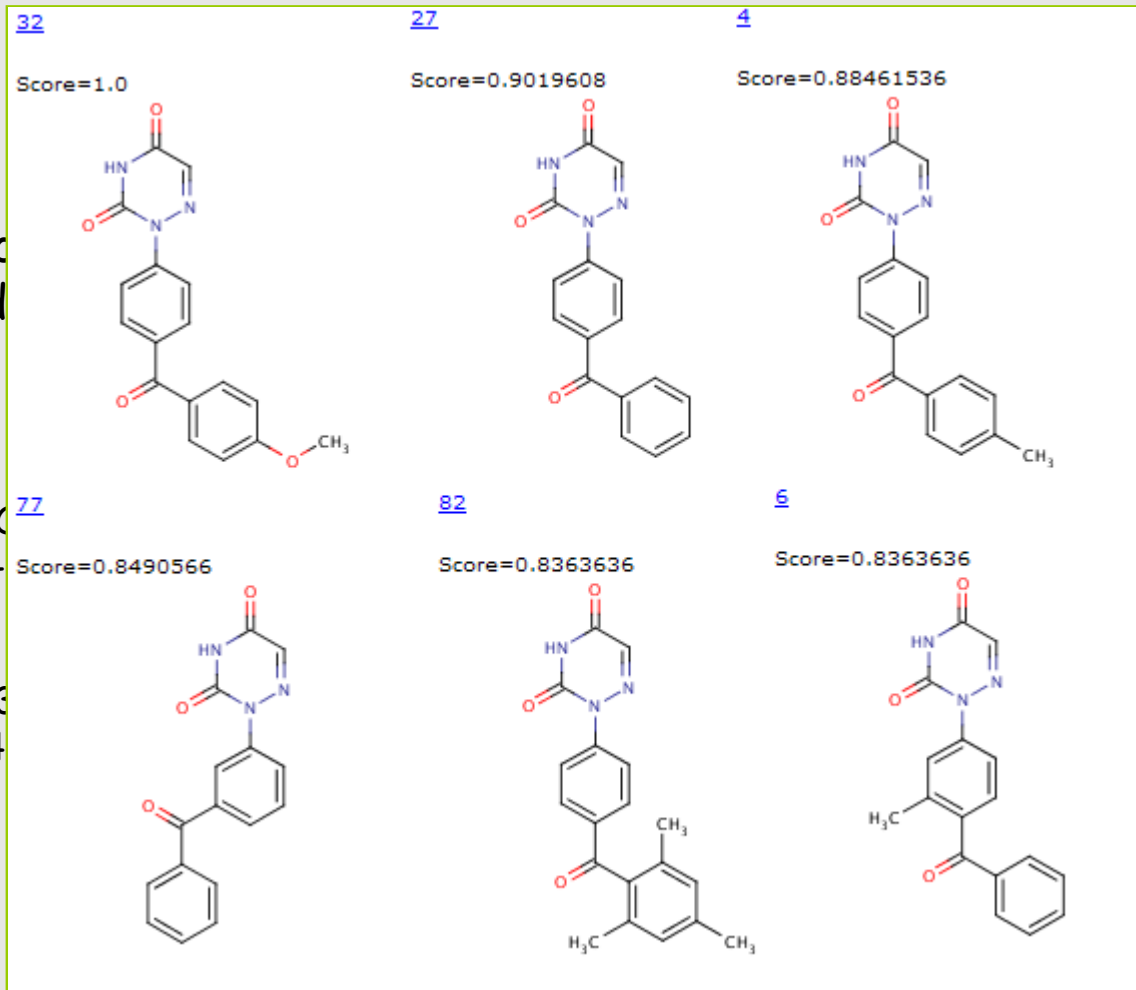
# OrChem similarity search example

Example: find (mol based) compounds *similar* to compound 32, with a minimum similarity score of 0.8

```
q6> select orchem_simsearch  
      from orchem_compound  
      where id = 32
```

```
ORCHEM_SIMSEARCH.SEARCH(MO
```

```
-----  
ORCHEM_COMPOUND_LIST(  
ORCHEM_COMPOUND('32', ' CDK  
ORCHEM_COMPOUND('27', ... , .983  
ORCHEM_COMPOUND('4', ... , .9384  
.... etc
```



# That's it ! Thank you for your attention

- And also thanks to
  - EBI Cheminformatics and Metabolism Team
  - CDK developers
  - EBI ChEMBL team