pgchem::tigress – A chemoinformatics extension for PostgreSQL: User Guide

Ernst-Georg Schmid

v.1.1 GiST – Winter 2009

Contents

Part I
Caveat

pgchem::tigress may contain errors in functionality and code. Therefore it should not be used unconsiderately and does not replace the advice of a trained chemist. Notably, pgchem::tigress was not designed to be used in GxP environments, for material safety systems and other safety critical environments.

Part II
Introduction to pgchem::tigress

1 Overview

pgchem::tigress is a chemoinformatics extension to the PostgreSQL object-relational database management system. It enables PostgreSQL to handle chemical datatypes. Pgchem::tigress is basically a wrapper around the checkmol/matchmol molecular analyzer and the OpenBabel computational chemistry package, plus some database functions, datatypes, a GiST index and auxiliary tables to access their functionality purely through SQL statements.

Pgchem::tigress supports exact and substructure searching on molecules, searching by functional groups, calculation of chemical properties like molecular formula and molecular weight and Tanimoto similarity searching.

 As of version 8.x of PostgreSQL, pgchem::tigress compiles and runs natively (at least) on Linux, Solaris, OS X and Win32.

2 Why PostgreSQL?

Because it is tried and tested, suitable for heavy-duty applications and has a clean interface for custom extensions. So far it has proven to be a good choice, but there are no reasons why there should not be a fbchem::tigress for Firebird or an orachem::tigress for Oracle in the future. Actually, mychem for MySQL has started at http://sourceforge.net/projects/mychem/.

Part III
Designing your schema

As of version 1.0 GiST, pgchem::tigress does not impose limits to your database design anymore. You can have as many molecules columns in as many tables

```
CREATE TABLE <moleculetable>
(
  <key> <type> NOT NULL,
  <moleculecolumn> molecule NOT NULL,
  CONSTRAINT pk_molecules PRIMARY KEY (<key>)
)
```

Figure 1: A basic molecules table

```
CREATE INDEX <indexname> ON <moleculetable>
USING GIST(<moleculecolumn>)
```

Figure 2: Creating the GiST index

as you need and have individual molecular indexes on each[1].

3 The molecules table

The molecules table holds the molecules itself, but can be extended to store whatever additional information should be attached to those molecules. It consists of at least two columns, as shown in Figure 1.

The actual name of the molecule column is unimportant for pgchem::tigress, choose it to your liking, but it has to be of type *molecule*. After the table has been created, a GiST index can be attached to the molecule column as shown in Figure 2.

4 The functional groups table

Optional since 1.0 GiST. It contains 0..n rows per molecule. As shown in Figure 3, each row contains a code that indicates the presence of a specific functional group in this molecule. Checkmol/matchmol currently detects 190 functional groups, thus does pgchem::tigress. This table can be used to directly search for

[1] An operator overwiew that shows which operators are GiST capable can be found in Table 4.

```
CREATE TABLE <molfgroupstable>
(
  <key> <type> NOT NULL,
  code char(8) NOT NULL,
  CONSTRAINT molfgroups_pkey PRIMARY KEY (<key>, code),
  CONSTRAINT fk_mol FOREIGN KEY (<key>) REFERENCES
  <moleculetable> (<key>)
  ON UPDATE NO ACTION ON DELETE CASCADE
)
```

Figure 3: The functional groups table

```
SELECT <moleculetable>.<key> FROM
<moleculetable> WHERE <querymolecule> = <moleculecolumn>;
```

Figure 4: Exact match

```
SELECT <moleculetable>.<key> FROM
<moleculetable> WHERE
<querymolecule> <= <moleculecolumn>;
```

Figure 5: Substructure match

molecules containing a given set of functional groups. See Part IV about how to do this in SQL.

Part IV
Working with data

If the Schema is set up correctly, start loading your molecules table by any means you like. The input to create a molecule is textual. The recognized formats are shown in Table 1. How loading code has to look like for a specific

Input format	**Coordinates**
MDL V2000 molfile	yes
MDL V3000 molfile	yes
SMILES	no
InChI	no

Table 1: The molecule input formats

programming language/database driver combination is beyond the scope of this manual.

5 Molecule searching

Exact searching can be done by using the = operator as shown in Figure 4. Substructure searching is done using the <= or >= operators. Figure 5 contains an example how to perform such a search. To search for molecules containing one or more functional groups, just select the desired functional group codes from the functional groups table and join it with the molecules table. The example scripts that come with pgchem::tigress contain a lookup table, containing all known codes with their english names. This can be used to search by names instead of codes.

Tanimoto similarity searching is done by the @ operator, which takes two molecules and returns their Tanimoto coefficient (Figure 6).

```
SELECT <moleculetable>.<key> FROM
<moleculetable> WHERE
<querymolecule> @ <moleculecolumn>) >= 0.9
```

Figure 6: Similarity search

6 Other molecule functions

- *SMARTSmatch(text,molecule)* takes a query SMARTS, a molecule to substructure match the query against. A return value of TRUE indicates a match. This function can be used as a postprocessor to the search operators to further refine your queries.
- *SMARTSmatch_count(text,molecule)* takes a query SMARTS, a molecule to substructure match the query against. The integer return value indicates how many unique matches were found in the target molecule. This function can be used as a postprocessor to the search operators to further refine your queries.
- *molkeys_long(molecule,bool,bool,bool)* takes a molecule and returns its fingerprint in long form. Long means, that for every column a name/value pair *name:value;* is generated. The first flag toggles strict checking of charges, the second flag toggles strict checking of isotopes and the the third flag toggles strict checking of radicals.
- *fgroup_codes(molecule)* takes a molecule and returns its functional group codes. This can be used to obtain the codes for a functional group search by drawn example.
- *fpmaccsstring(molecule)* takes a molecule and returns its binary MACCS fingerprint.

7 Calculating properties

- *molweight(molecule)* takes a molecule and returns the standard molar mass given by IUPAC atomic masses, including all implicit hydrogens.
- *exactmass(molecule)* takes a molecule and returns the the mass given by isotopes (or most abundant isotope, if not specified), including all implicit hydrogens .
- *total_charge(molecule)* takes a molecule and returns the total charge (0=neutral), including all implicit hydrogens.
- *number_of_atoms(molecule)* takes a molecule and returns the number of atoms, including all implicit hydrogens.
- *number_of_heavytoms(molecule)* takes a molecule and returns the number of heavy atoms. This also counts Deuterium an Tritium!
- *number_of_bonds(molecule)* takes a molecule and returns the number of bonds, including all implicit hydrogens.

- *number_of_rotatable_bonds(molecule)* takes a molecule and returns the number of rotatable bonds[2].
- *is_chiral(molecule)* takes a molecule and tries to perceive its chirality.
- *is_2D(molecule)* takes a molecule and returns true if 2D coordinates are present.
- *is_3D(molecule)* takes a molecule and returns true if 3D coordinates are present.
- *molformula(molecule)* takes a molecule and returns the molformula.
- *logP(molecule)* takes a molecule and returns the predicted log P value.
- *MR(molecule)* takes a molecule and returns the predicted molar refractivity.
- *PSA(molecule)* takes a molecule and returns the predicted polar surface area.

8 Conversions

- *migrate_molecule(bytea)* migrates a < 1.0 GiST molecule to >= 1.0 GiST.
- *v3000(molecule)* takes a molecule and converts it to a V3000 molfile.
- *smiles(molecule,bool)* takes a molecule and converts it to a SMILES string. Parameter two controls if all isotopic or chiral markings shall be omitted.
- *canonical_smiles(molecule)* takes a molecule and converts it to a canonical SMILES string.
- *inchi(molecule)* takes a molecule and converts it to a IUPAC InChI string.
- *inchikey(molecule)* takes a molecule and converts it to a IUPAC InChI-key string.

9 Manipulation

- *strip_salts(molecule,bool)* takes a molecule and strips all atoms except for the largest contiguous fragment. If the second parameter is true, the charge(s) of the result are neutralized.
- *add_hydrogens(molecule,bool,bool)* takes a molecule and adds hydrogens. Parameter one controls if all or only polar hydrogens are added and parameter two if a correction for PH=7.4 shall be done.
- *remove_hydrogens(molecule,bool)* takes a molecule and removes hydrogens. Parameter two controls if all or only non-polar hydrogens (true) shall be removed. This also removes Deuterium and Tritium.

[2]Any non-ring bond with hybridization of sp2 or sp3 is considered a potentially rotatable bond. There is no special bond-typing, e.g. for amide C-N bonds with their high rotational energy barrier.

10 The Lipinsky filter

The Lipinsky filter function *lipinsky(molecule)* checks a molecule against the Lipinsky criteria.

Criterion match	**Output letter**
none	empty string
H donors > 5	A
molecular weight > 500	B
log P > 5.0	C
H acceptors > 10	D

Table 2: The Lipinsky function output format

The output is either a string consisting of any combination of the letters A, B, C and D or an empty string, as shown in Table 2.

11 Helper functions

- *validate_cas_no(varchar)* takes a CAS-No. and checks its validity with the official CAS checksum algorithm, including the 10 digit CAS-Numbers introduced in 2008.
- *is_nostruct(molecule)* checks if a molecule is a MDL NoStruct.
- *disconnected(molecule)* checks if a molecule is disconnected.
- *pgchem_version()* returns the pgchem::tigress version identifier.
- *pgchem_barsoi_version()* returns the barsoi version identifier.

Part V
Miscellaneous

12 Rejecting duplicate molecules

In order to emulate a unique constraint on molecules, a row level `INSERT` and `UPDATE` trigger can be used. First create a trigger function like that in Figure 7. As the exact search in this function is dependent on the name and layout of the specific molecules table, do not put this in the public schema. Then attach a trigger like Figure 8 to that molecule table. Every new molecule will now be compared to those already in the table and rejected if it is a duplicate.

13 Tuning

- Put GiST indexes on the molecule columns.
- Frequently update the statistics on the tables.

```
CREATE OR REPLACE FUNCTION example.t_is_molecule_unique()
  RETURNS "trigger" AS
$BODY$
DECLARE is_not_unique bool;
BEGIN

is_not_unique:=false;

IF TG_OP='INSERT' OR TG_OP='UPDATE' THEN

is_not_unique := EXISTS (SELECT <key> FROM <moleculetable> WHERE
NEW.<moleculecolumn> = <moleculecolumn>);

  IF is_not_unique THEN  RAISE EXCEPTION 'MOLECULE IS NOT
  UNIQUE!'; END IF;

ELSE

  RAISE EXCEPTION 'PGCHEM IS-MOLECULE-UNIQUE TRIGGER CALLED
  OUTSIDE INSERT OR UPDATE!';

END IF;
RETURN NEW;
END;
$BODY$
LANGUAGE 'plpgsql' VOLATILE;
```

Figure 7: The unique molecules trigger function

- Configure PostgreSQL correctly for your type and size of application.
- Query the database as precise as possible, especially for substructure searches, e.g. avoid to search for Benzene or Naphthalene as substructures without further constraints.
- Use the LIMIT option for PostgreSQL queries if you want to limit the number of hits returned. LIMIT kills the entire query at once when the specified result set limit has been reached, effectively reducing the workload for high-yield queries. LIMIT does not work together with SELECT COUNT.

```
CREATE TRIGGER is_molecule_unique
  BEFORE INSERT OR UPDATE
  ON <moleculetable>
  FOR EACH ROW
  EXECUTE PROCEDURE example.t_is_molecule_unique();
```

Figure 8: The unique molecules trigger

- Pgchem::tigress is generally a bit faster on UN*X than on Win32.

14 Security

- Whitelist-validating all data in an application on input and output is always a good idea.
- pgchem::tigress and barsoi have be hardened against classical buffer overflows, but may not be immune.

15 Limitations

- MDL NoStructs are very weakly supported. Avoid them if you can. The concept of a special non-structure is grotesque anyway when you can use NULL.
- Disconnected structures cannot be used as query structures. However, they can be search targets.

16 Links & Things

- checkmol/matchmol: http://merian.pch.univie.ac.at/~nhaider/cheminf/cmmm.html
- OpenBabel: http://openbabel.sourceforge.net/
- PostgreSQL: http://www.postgresql.org/
- pgchem::tigress + barsoi: http://pgfoundry.org/projects/pgchem/

A The checkmol/matchmol molecular keys format

See Table 3.

B The operators

See Table 4.

Field(s)	Descriptor
ntoms, n_bonds, n_rings	number of atoms, bonds, rings
n_QA, n_QB, n_chg	number of query atoms, query bonds, charges
n_C1, n_C2, n_C	number of sp, sp2 hybridized, and total no. of carbons
n_CHB1p, n_CHB2p, n_CHB3p, n_CHB4	number of C atoms with at least 1, 2, 3 hetero bonds
n_O2, n_O3	number of sp2 and sp3 oxygens
n_N1, n_N2, n_N3	number of sp, sp2, and sp3 nitrogens
n_S, n_SeTe	number of sulfur atoms and selenium/tellurium atoms
n_F, n_Cl, n_Br, n_I	number of fluorine, chlorine, bromine, iodine atoms
n_P, n_B	number of phosphorus and boron atoms
n_Met, n_X	number of metal and "other" atoms (not listed elsewhere)
n_b1, n_b2, n_b3, n_bar	number single, double, triple, and aromatic bonds
n_C1O, n_C2O, n_CN, n_XY	number of C-O single bonds, C=O double bonds, CN bonds (any type), hetero/hetero bonds
n_r3, n_r4, n_r5, n_r6, n_r7, n_r8	number of 3-, 4-, 5-, 6-, 7-, and 8-membered rings
n_r9, n_r10, n_r11, n_r12, n_r13p	number of 9-, 10-, 11-, 12-, and 13plus-membered rings
n_rN, n_rN1, n_rN2, n_rN3p	number of rings containing N (any number), 1 N, 2 N, and 3 N or more
n_rO, n_rO1, n_rO2p	number of rings containing O (any number), 1 O, and 2 O or more
n_rS, n_rX, n_rAr, n_rBz	number of rings containing S (any number), any heteroatom (any number), number of aromatic rings, number of benzene rings
n_br2p	number of bonds belonging to more than one ring
n_psg01, n_psg02, n_psg13, n_psg14	number of atoms belonging to elements of group 1, 2, etc.
n_psg15, n_psg16, n_psg17, n_psg18	number of atoms belonging to elements of group 15, 16, etc.
n_pstm, n_psla	number of transition metals, lanthanides/actinides
n_iso, n_rad	number of isotopes, radicals

Table 3: The checkmol/matchmol molecular keys format, all descriptors are integers

Operator	Description	GiST acceleration	Return type
A <= B	A contained in B	yes	boolean
A >= B	A contains B	yes	boolean
A = B	A equals B	yes	boolean
A @ B	Tanimoto coefficient of A and B	no	double

Table 4: The available operators for the molecule datatype