

CEFIC LRI RfP EEM-9
**Building blocks for a future
(Q)SAR Decision Support System:
Databases, applicability domain and structure conversions**
(Codename AMBIT)

Ambit Database Schema and API

Nina Nikolova-Jeliazkova
IPP - Bulgarian Academy of Science, Sofia, Bulgaria

Joanna Jaworska
P&G Brussels, Belgium

April 7, 2005

1.Introduction

1.1.Background

This report concerns mostly EEM9- 3 “Golden” database. The objectives are

1. IT: To develop (design and codify) well documented database schema and API (application programming interface) and use it in this and the EEM9-1a, EEM9-1b, EEM9-2 parts of the proposal
2. Data collection: Import already available quality checked data in a computer database.

This report describes the implementation of the first objective. The data collection is planned for the further phase of the project.

The “database” in an IT sense is a software system, allowing users to store, to retrieve and to search structured information. In this project we need to store chemical information (structure, names, CAS, SMILES, properties, etc.), in order to provide requested functionalities. The database is a supporting tool for the EEM9-1a Applicability Domain, EEM9-1b Chemical Grouping and EEM9-2 CAS-SMILES converter.

The software developed is based on a Relational Database Management System, which allows much faster and convenient access to the data in contrast to flat text files. In particular, we use MySQL (www.mysql.com), which is known as the most popular open source relational database.

1.2.Codename origin

A major part of the project is concerned with the applicability domain of QSAR models, i.e. how to determine whether a QSAR model can be applied for a given chemical compound. Since it means finding the scope of a QSAR model, the working name of the project became **AMBIT**¹

¹ A dictionary entry for the word "ambit":

Definition: [n] an area in which something acts or operates or has power or control: "the range of a supersonic jet"; "the ambit of municipal legislation"; "within the compass of this article"; "within the scope of an investigation"; "outside the reach of the law"; "in the political orbit of a world power"

Synonyms: compass, orbit, range, reach, scope

Building blocks for QSAR Decision Support System

2.Database schema

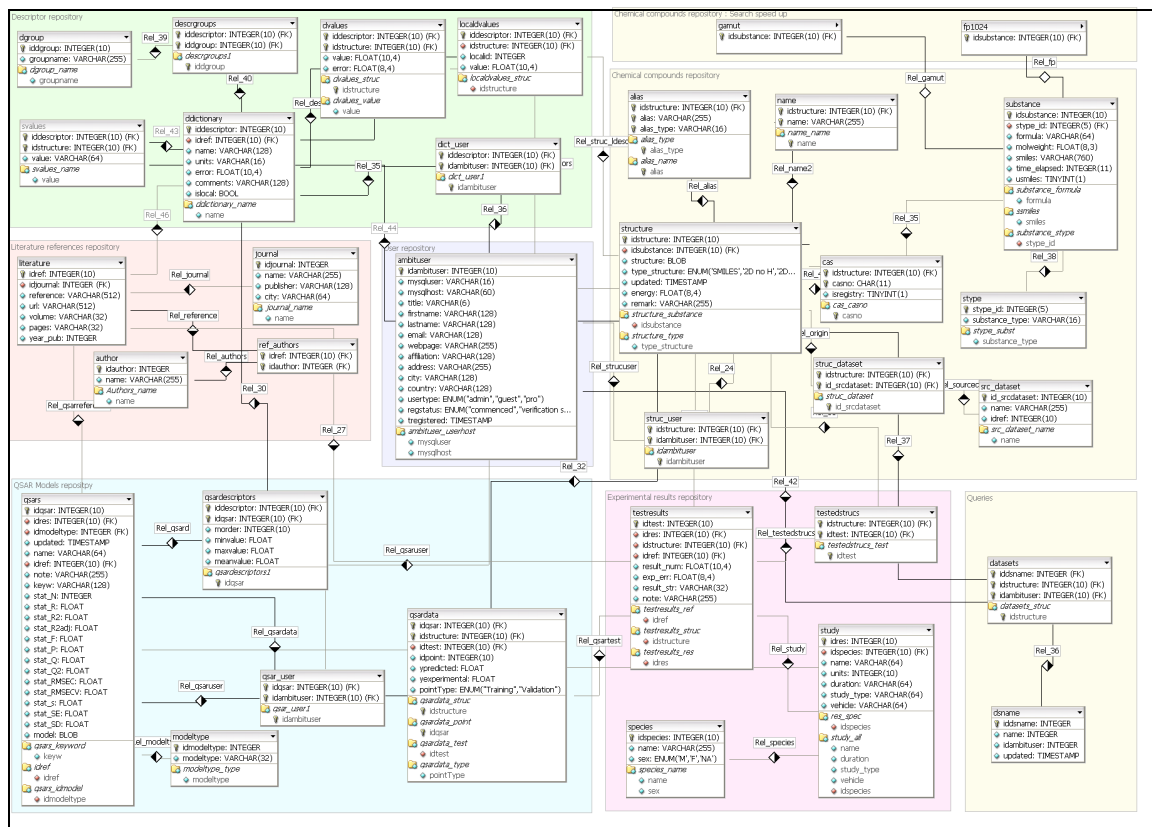


Figure 1. Ambit database structure

The database schema (Figure 1) contains 33 tables and could be divided in 6 components:

Component	Color on Figure 1	Tables
User repository	blue	1
Literature references repository	red	4
Experimental results repository	magenta	4
QSAR models repository	cyan	5
Descriptors repository	green	6
Chemical compounds repository	yellow	13
All		33

Building blocks for QSAR Decision Support System

2.1. User repository

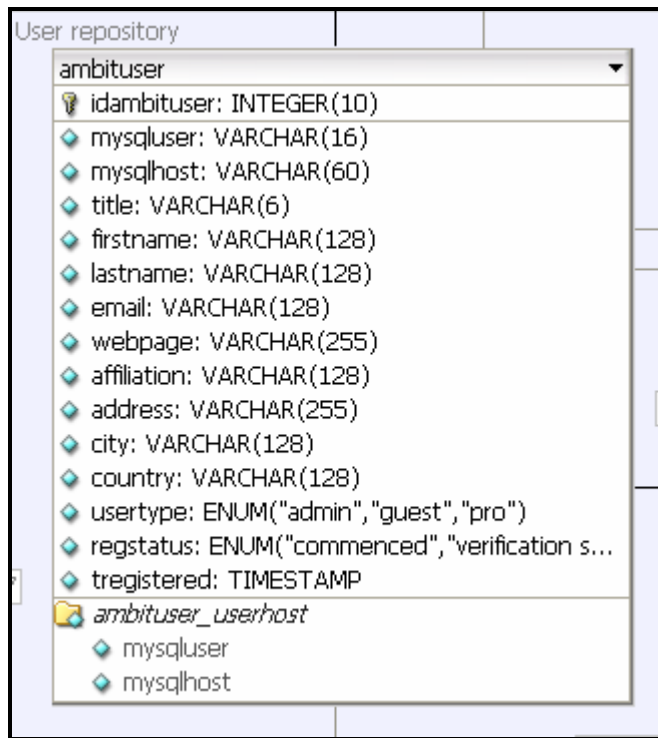


Figure 2. User table

The user have to connect and log in into the database in order to use it. Only registered AMBIT users are allowed to log in.

For each structure, descriptor and model, the user id of the user entering the information is recorder in the corresponding tables.

Table ambituser
-- The table ambituser stores information about the users who have the right to login and use AMBIT database. -- There are three different kind of users : guest, admin and pro. Guest users are allowed only to query the database, biut not to modify/insert records. Pro users are allowed to modify and add new data. Admin users are allowed to do everything, including deleting and creating the tables. -- Passwords are not stored here, the standard mysql users are used to hold user/password details.
CREATE TABLE ambituser (idambituser INTEGER(10) UNSIGNED NOT NULL, mysqluser VARCHAR(16) NOT NULL, mysqlhost VARCHAR(60) NOT NULL, title VARCHAR(6) NOT NULL DEFAULT "",

Building blocks for QSAR Decision Support System

```

    firstname VARCHAR(128) NOT NULL,
    lastname VARCHAR(128) NOT NULL,
    email VARCHAR(128) NOT NULL,
    webpage VARCHAR(255) NOT NULL DEFAULT "",
    affiliation VARCHAR(128) NOT NULL DEFAULT "",
    address VARCHAR(255) NOT NULL DEFAULT "",
    city VARCHAR(128) NOT NULL DEFAULT "",
    country VARCHAR(128) NOT NULL DEFAULT "",
    usertype ENUM("admin","guest","pro") NOT NULL DEFAULT "guest",
    regstatus ENUM("commenced","verification sent","verified") NOT NULL DEFAULT
"commenced",
    tregistered TIMESTAMP NOT NULL,
    PRIMARY KEY(idambituser),
    UNIQUE INDEX ambituser_userhost(mysqluser, mysqlhost)
)
TYPE=InnoDB;
insert          ignore          into          ambituser
(mysqluser,mysqlhost,firstname,lastname,email,regstatus,usertype)
values ("guest","%","guest","user","nina@acad.bg","verified","guest");

```

2.2. Literature references repository

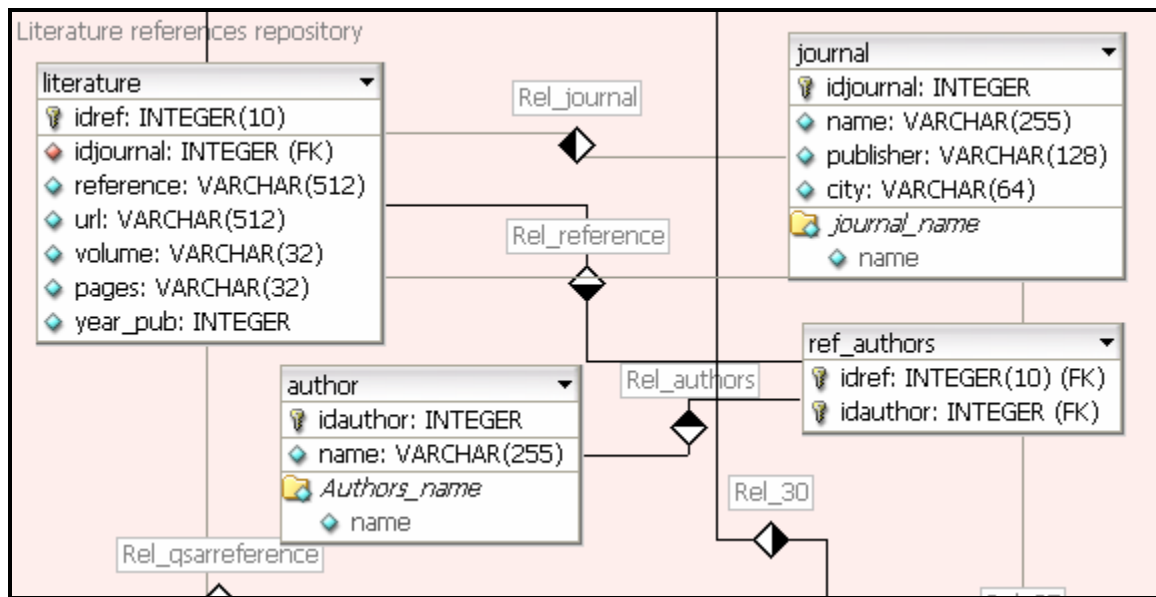


Figure 3. Literature reference tables

Building blocks for QSAR Decision Support System

Literature references are used when storing compounds, descriptors, models and experimental data.

Table journal
-- This table stores information about journals. Referenced by table literature.
CREATE TABLE journal (idjournal INTEGER UNSIGNED NOT NULL AUTO_INCREMENT, name VARCHAR(255) NOT NULL, publisher VARCHAR(128) NULL, city VARCHAR(64) NULL, PRIMARY KEY(idjournal), UNIQUE INDEX journal_name (name)) TYPE=InnoDB;

Table author
-- This table stores author names. Referenced by table literature and ref_authors.
CREATE TABLE author (idauthor INTEGER UNSIGNED NOT NULL AUTO_INCREMENT, name VARCHAR(255) NOT NULL, PRIMARY KEY(idauthor), UNIQUE INDEX Authors_name (name)) TYPE=InnoDB;

Table literature
-- This table stores information of a literature reference.
CREATE TABLE literature (idref INTEGER(10) UNSIGNED NOT NULL AUTO_INCREMENT, idjournal INTEGER UNSIGNED NOT NULL, reference VARCHAR(512) NOT NULL, url VARCHAR(512) NOT NULL, volume VARCHAR(32) NULL, pages VARCHAR(32) NULL, year_pub INTEGER UNSIGNED NULL, PRIMARY KEY(idref), FOREIGN KEY(idjournal) REFERENCES journal(idjournal) ON DELETE CASCADE ON UPDATE CASCADE) TYPE=InnoDB;

Building blocks for QSAR Decision Support System

Table ref_authors
-- This table stores the link between the authors and literature reference.
<pre>CREATE TABLE ref_authors (idref INTEGER(10) UNSIGNED NOT NULL, idauthor INTEGER UNSIGNED NOT NULL, PRIMARY KEY(idref, idauthor), FOREIGN KEY(idref) REFERENCES literature(idref) ON DELETE CASCADE ON UPDATE CASCADE, FOREIGN KEY(idauthor) REFERENCES author(idauthor) ON DELETE CASCADE ON UPDATE CASCADE) TYPE=InnoDB;</pre>

Figure 4. SQL script to create the literature reference repository

2.3. Experimental results repository

Building blocks for QSAR Decision Support System

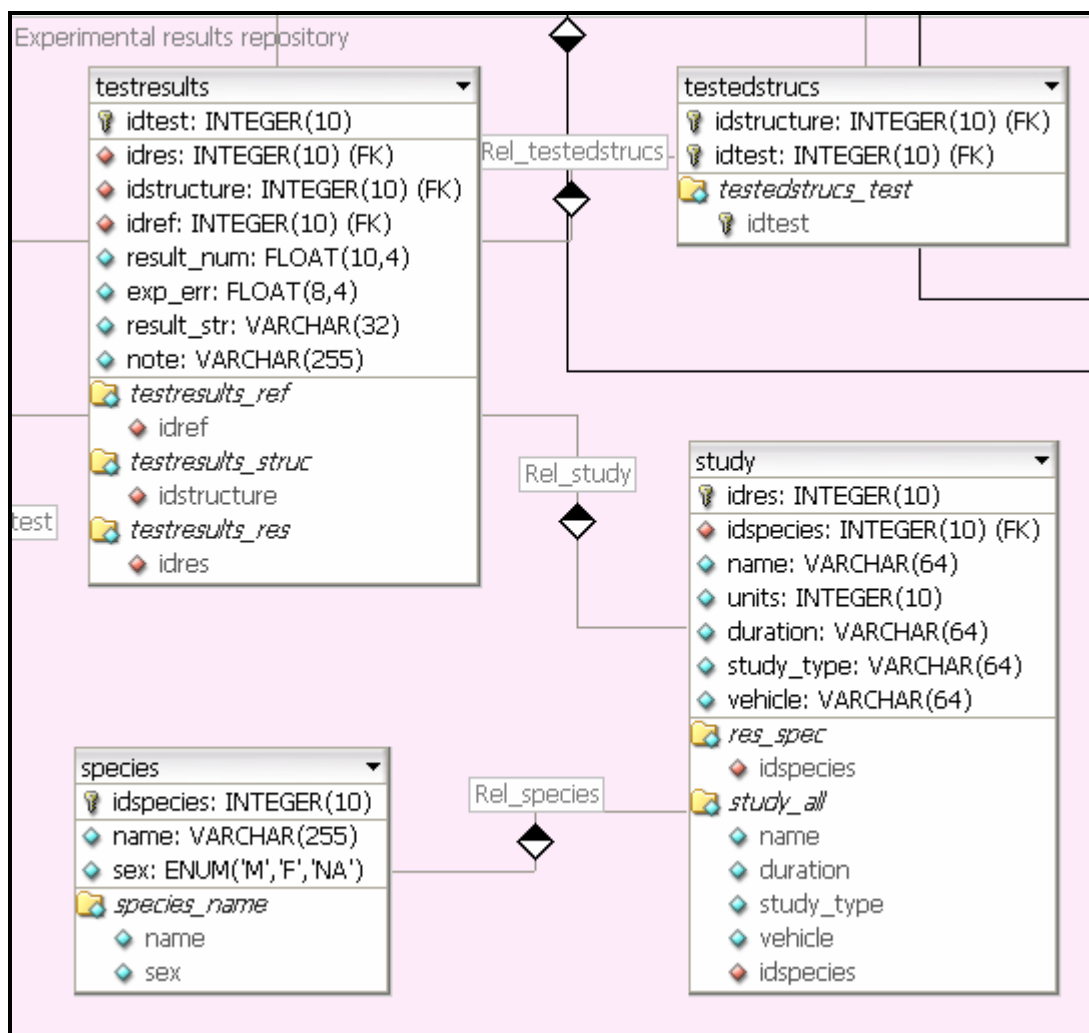


Figure 5. Experiment results repository

Table species
-- Species information
<pre> CREATE TABLE species (idspecies INTEGER(10) UNSIGNED NOT NULL AUTO_INCREMENT, name VARCHAR(255) NOT NULL DEFAULT "NA", sex ENUM('M','F','NA') NOT NULL DEFAULT "NA", PRIMARY KEY(idspecies), UNIQUE INDEX species_name(name, sex)) </pre>
TYPE=InnoDB;

Building blocks for QSAR Decision Support System

Table study
-- Information about the type of the study. Each row consists of an unique autogenerated identifier (idres) link to the species (idspecies), name, units, duration, study_type, vehicle. -- If a field is not relevant, or the information is not supplied then the default value "NA" is stored.
CREATE TABLE study (idres INTEGER(10) UNSIGNED NOT NULL AUTO_INCREMENT, idspecies INTEGER(10) UNSIGNED NOT NULL, name VARCHAR(64) NOT NULL DEFAULT "NA", units INTEGER(10) UNSIGNED NOT NULL, duration VARCHAR(64) NOT NULL DEFAULT "NA", study_type VARCHAR(64) NOT NULL DEFAULT "NA", vehicle VARCHAR(64) NOT NULL DEFAULT "NA", PRIMARY KEY(idres), INDEX res_spec(idspecies), UNIQUE INDEX study_all(name, duration, study_type, vehicle, idspecies), FOREIGN KEY(idspecies) REFERENCES species(idspecies) ON DELETE RESTRICT ON UPDATE CASCADE) TYPE=InnoDB;

Table testedstrucs
-- This is to contain the actual structure tested, if it differs from the simplified one in testresults.
CREATE TABLE testedstrucs (idstructure INTEGER(10) UNSIGNED NOT NULL, idtest INTEGER(10) UNSIGNED NOT NULL, PRIMARY KEY(idstructure, idtest), INDEX testedstrucs_test(idtest), FOREIGN KEY(idtest) REFERENCES testresults(idtest) ON DELETE RESTRICT ON UPDATE CASCADE, FOREIGN KEY(idstructure) REFERENCES structure(idstructure) ON DELETE RESTRICT ON UPDATE CASCADE) TYPE=InnoDB;

Table testresults
-- This table stores information about experimental results. Each row consists of a link to the species (idres), link to the structure (idstructure), link to the literature reference (idref) and experimental results itself. The field result_str is provided in order to store a string representation of an

Building blocks for QSAR Decision Support System

experimental results , if necessary. The "note" field may contain free text comment.

```
CREATE TABLE testresults (  
  idtest INTEGER(10) UNSIGNED NOT NULL AUTO_INCREMENT,  
  idres INTEGER(10) UNSIGNED NOT NULL,  
  idstructure INTEGER(10) UNSIGNED NOT NULL,  
  idref INTEGER(10) UNSIGNED NOT NULL,  
  result_num FLOAT(10,4) NOT NULL DEFAULT 0.0000,  
  exp_err FLOAT(8,4) NOT NULL DEFAULT 0.0000,  
  result_str VARCHAR(32) NOT NULL DEFAULT "",  
  note VARCHAR(255) NOT NULL DEFAULT "",  
  PRIMARY KEY(idtest),  
  INDEX testresults_ref(idref),  
  INDEX testresults_struc(idstructure),  
  INDEX testresults_res(idres),  
  FOREIGN KEY(idref)  
    REFERENCES literature(idref)  
    ON DELETE RESTRICT  
    ON UPDATE CASCADE,  
  FOREIGN KEY(idstructure)  
    REFERENCES structure(idstructure)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  FOREIGN KEY(idres)  
    REFERENCES study(idres)  
    ON DELETE RESTRICT  
    ON UPDATE CASCADE  
)  
TYPE=InnoDB;
```

Figure 6. SQL script to create the experimental results repository

Building blocks for QSAR Decision Support System

```
idqsar INTEGER(10) UNSIGNED NOT NULL,  
idstructure INTEGER(10) UNSIGNED NOT NULL,  
idtest INTEGER(10) UNSIGNED NOT NULL,  
idpoint INTEGER(10) UNSIGNED NOT NULL,  
ypredicted FLOAT NOT NULL,  
yexperimental FLOAT NOT NULL,  
pointType ENUM("Training","Validation") NOT NULL,  
PRIMARY KEY(idqsar, idstructure),  
INDEX qsardata_struct(idstructure),  
INDEX qsardata_point(idqsar),  
INDEX qsardata_test(idtest),  
INDEX qsardata_type(pointType),  
FOREIGN KEY(idqsar)  
  REFERENCES qsars(idqsar)  
  ON DELETE CASCADE  
  ON UPDATE CASCADE,  
FOREIGN KEY(idstructure)  
  REFERENCES structure(idstructure)  
  ON DELETE RESTRICT  
  ON UPDATE CASCADE,  
FOREIGN KEY(idtest)  
  REFERENCES testresults(idtest)  
  ON DELETE RESTRICT  
  ON UPDATE CASCADE  
)  
TYPE=InnoDB;
```

Table **qsardescriptors**

```
-- This table stores a list of descriptor identifiers (iddescriptor, a link to  
ddictionary table) , used in a particular qsar model (idqsar, a link to qsars  
table). Morder is the order of the descriptor in the model, i.e. first,  
second, etc.  
-- minvalue, maxvalue and meanvalue are only for reference and are not  
mandatory.
```

```
CREATE TABLE qsardescriptors (  
  iddescriptor INTEGER(10) UNSIGNED NOT NULL,  
  idqsar INTEGER(10) UNSIGNED NOT NULL,  
  morder INTEGER(10) UNSIGNED NOT NULL DEFAULT 0,  
  minvalue FLOAT NULL,  
  maxvalue FLOAT NULL,  
  meanvalue FLOAT NULL,  
  PRIMARY KEY(iddescriptor, idqsar),  
  INDEX qsardescriptors1(idqsar),  
  FOREIGN KEY(iddescriptor)  
    REFERENCES ddictionary(iddescriptor)  
    ON DELETE RESTRICT  
    ON UPDATE CASCADE,  
  FOREIGN KEY(idqsar)  
    REFERENCES qsars(idqsar)  
    ON DELETE CASCADE
```

Building blocks for QSAR Decision Support System

```
        ON UPDATE CASCADE
    )
TYPE=InnoDB;
```

Table **qsars**

```
-- This is the table that contains QSAR models themselves.  Each row has an
unique autogenerated identifier idqsar , an "update" field to store when the
model was inserted, a name , a list of keywords and a free text note.
-- "idmodeltype" is a link to the model type.
-- "idref" is a link to the literature reference.
-- "idres" is a link to the study used in this model.  If data from many studies
has been combined in a single model, then idres could point to an empty study.
-- Fields to hold all kind of statistics follow.
-- The model itself is to be stored in the "model" blob field.  Which format is
to be used is still to be decided, but perhaps some kind of XML.
```

```
CREATE TABLE qsars (
  idqsar INTEGER(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  idres INTEGER(10) UNSIGNED NOT NULL,
  idmodeltype INTEGER UNSIGNED NOT NULL,
  updated TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  name VARCHAR(64) NOT NULL,
  idref INTEGER(10) UNSIGNED NOT NULL,
  note VARCHAR(255) NOT NULL,
  keyw VARCHAR(128) NOT NULL,
  stat_N INTEGER UNSIGNED NULL,
  stat_R FLOAT NULL,
  stat_R2 FLOAT NULL,
  stat_R2adj FLOAT NULL,
  stat_F FLOAT NULL,
  stat_P FLOAT NULL,
  stat_Q FLOAT NULL,
  stat_Q2 FLOAT NULL,
  stat_RMSEC FLOAT NULL,
  stat_RMSECV FLOAT NULL,
  stat_s FLOAT NULL,
  stat_SE FLOAT NULL,
  stat_SD FLOAT NULL,
  model BLOB NULL,
  PRIMARY KEY(idqsar),
  INDEX qsars_keyword(keyw),
  INDEX idref(idref),
  INDEX qsars_idmodel(idmodeltype),
  FOREIGN KEY(idref)
  REFERENCES literature(idref)
  ON DELETE RESTRICT
```

Building blocks for QSAR Decision Support System

```
        ON UPDATE CASCADE,  
FOREIGN KEY(idmodeltype)  
  REFERENCES modeltype(idmodeltype)  
  ON DELETE NO ACTION  
  ON UPDATE CASCADE,  
FOREIGN KEY(idres)  
  REFERENCES study(idres)  
  ON DELETE NO ACTION  
  ON UPDATE NO ACTION  
)  
TYPE=InnoDB;
```

Table qsar_user
-- This is to keep track which user had been inserted each model
<pre>CREATE TABLE qsar_user (idqsar INTEGER(10) UNSIGNED NOT NULL, idambituser INTEGER(10) UNSIGNED NOT NULL, PRIMARY KEY(idqsar, idambituser), INDEX qsar_user1(idambituser), FOREIGN KEY(idqsar) REFERENCES qsars(idqsar) ON DELETE CASCADE ON UPDATE CASCADE, FOREIGN KEY(idambituser) REFERENCES ambituser(idambituser) ON DELETE RESTRICT ON UPDATE CASCADE) TYPE=InnoDB;</pre>

Figure 8. SQL script to create the QSAR models repository

2.5.Descriptors repository

Descriptors repository is where descriptor names and values are stored. There is a descriptor dictionary, where descriptor names are defined, along with literature references, units and comments. Local, global and string values can be stored in corresponding tables (see below).

Building blocks for QSAR Decision Support System

Additionally, a table to hold descriptor group names is defined. Group names are not predefined and are to be entered by users. A descriptor may belong to any number of groups.

Finally, the information about which user has inserted each particular descriptor is stored.

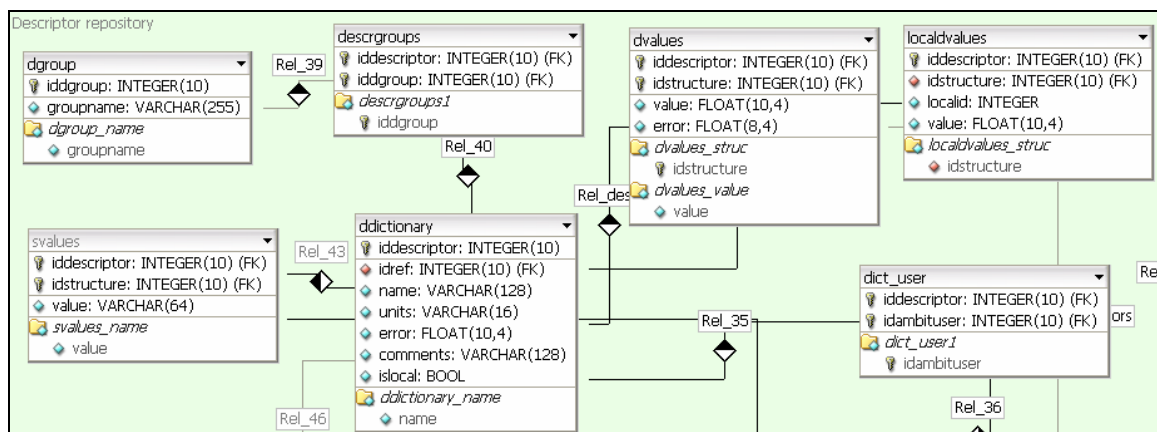


Figure 9. Descriptors repository

Table ddictionary
<pre>-- This is a descriptor dictionary. Each row stores a single descriptor with its unique autogenerated identifier "iddescriptor" and an unique name. -- "idref" is a link to a literature reference where this descriptor is defined (hopefully first). -- The fields to store units, default error (if any) , free text comments and a flag whether it is a local or global descriptor are provided.</pre>
<pre>CREATE TABLE ddictionary (iddescriptor INTEGER(10) UNSIGNED NOT NULL AUTO_INCREMENT, idref INTEGER(10) UNSIGNED NOT NULL, name VARCHAR(128) NOT NULL, units VARCHAR(16) NOT NULL, error FLOAT(10,4) NOT NULL DEFAULT 0.0000, comments VARCHAR(128) NOT NULL DEFAULT "", islocal BOOL NOT NULL DEFAULT "False", PRIMARY KEY(iddescriptor), UNIQUE INDEX ddictionary_name(name), FOREIGN KEY(idref) REFERENCES literature(idref) ON DELETE NO ACTION ON UPDATE NO ACTION) TYPE=InnoDB;</pre>

Building blocks for QSAR Decision Support System

Table descrgroups
-- This table indicates which descriptor belongs to which group. A descriptor may belong to any number of groups. See table dgroup.
<pre>CREATE TABLE descrgroups (iddescriptor INTEGER(10) UNSIGNED NOT NULL, iddgroup INTEGER(10) UNSIGNED NOT NULL, PRIMARY KEY(iddescriptor, iddgroup), INDEX descrgroups1(iddgroup), FOREIGN KEY(iddgroup) REFERENCES dgroup(iddgroup) ON DELETE CASCADE ON UPDATE CASCADE, FOREIGN KEY(iddescriptor) REFERENCES ddictionary(iddescriptor) ON DELETE CASCADE ON UPDATE CASCADE);</pre>

Table dgroup
-- This table stores a list of descriptor groups (i.e. topological, quantum mechanical, etc.). The list is not predefined and any group name can be added, provided that the name is unique. -- A descriptor may belong to any number of groups, thus making possible to use many different classifications existing in QSAR community ;)
<pre>CREATE TABLE dgroup (iddgroup INTEGER(10) UNSIGNED NOT NULL AUTO_INCREMENT, groupname VARCHAR(255) NOT NULL, PRIMARY KEY(iddgroup), UNIQUE INDEX dgroup_name(groupname)) TYPE=InnoDB;</pre>

Table dict_user
-- This is to keep track which descriptor was entered / modified by which user.
<pre>CREATE TABLE dict_user (iddescriptor INTEGER(10) UNSIGNED NOT NULL, idambituser INTEGER(10) UNSIGNED NOT NULL, PRIMARY KEY(iddescriptor, idambituser), INDEX dict_user1(idambituser), FOREIGN KEY(iddescriptor) REFERENCES ddictionary(iddescriptor) ON DELETE CASCADE ON UPDATE NO ACTION, FOREIGN KEY(idambituser) REFERENCES ambituser(idambituser) ON DELETE RESTRICT</pre>

Building blocks for QSAR Decision Support System

```
        ON UPDATE CASCADE
    )
TYPE=InnoDB;
```

Table **dvalues**

```
-- This itable stores the actual descriptor values. Each row points to a value
of descriptor "iddescriptor" for the structure "idstructure"
```

```
CREATE TABLE dvalues (
    iddescriptor INTEGER(10) UNSIGNED NOT NULL,
    idstructure INTEGER(10) UNSIGNED NOT NULL,
    value FLOAT(10,4) NOT NULL,
    error FLOAT(8,4) NOT NULL DEFAULT 0.0000,
    PRIMARY KEY(iddescriptor, idstructure),
    INDEX dvalues_struc(idstructure),
    INDEX dvalues_value(value),
    FOREIGN KEY(iddescriptor)
        REFERENCES ddictionary(iddescriptor)
        ON DELETE RESTRICT
        ON UPDATE CASCADE,
    FOREIGN KEY(idstructure)
        REFERENCES structure(idstructure)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
)
TYPE=InnoDB;
```

Table **localdvalues**

```
-- This is similar to dvalues, but used to store values for local descriptors.
```

```
CREATE TABLE localdvalues (
    iddescriptor INTEGER(10) UNSIGNED NOT NULL AUTO_INCREMENT,
    idstructure INTEGER(10) UNSIGNED NOT NULL,
    localid INTEGER UNSIGNED NOT NULL,
    value FLOAT(10,4) NOT NULL,
    PRIMARY KEY(iddescriptor),
    INDEX localdvalues_struc(idstructure),
    FOREIGN KEY(iddescriptor)
        REFERENCES ddictionary(iddescriptor)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY(idstructure)
        REFERENCES structure(idstructure)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
)
TYPE=InnoDB;
```

Building blocks for QSAR Decision Support System

Table svalues
-- This table stores string values for a structure, different than registry numbers and aliases. -- Example : a value "Narcotic" could be stored for a certain structure and a descriptor defined in the ddictionary table with the name "MOA type".
CREATE TABLE svalues (iddescriptor INTEGER(10) UNSIGNED NOT NULL, idstructure INTEGER(10) UNSIGNED NOT NULL, value VARCHAR(64) NOT NULL, PRIMARY KEY(iddescriptor, idstructure), INDEX svalues_name(value), FOREIGN KEY(iddescriptor) REFERENCES ddictionary(iddescriptor) ON DELETE NO ACTION ON UPDATE NO ACTION, FOREIGN KEY(idstructure) REFERENCES structure(idstructure) ON DELETE NO ACTION ON UPDATE NO ACTION) TYPE=InnoDB;

Figure 10. SQL script to create the descriptors repository

2.6. Chemical compounds repository

2.6.1. Chemical structures repository.

These are tables which store chemical compounds.

Building blocks for QSAR Decision Support System

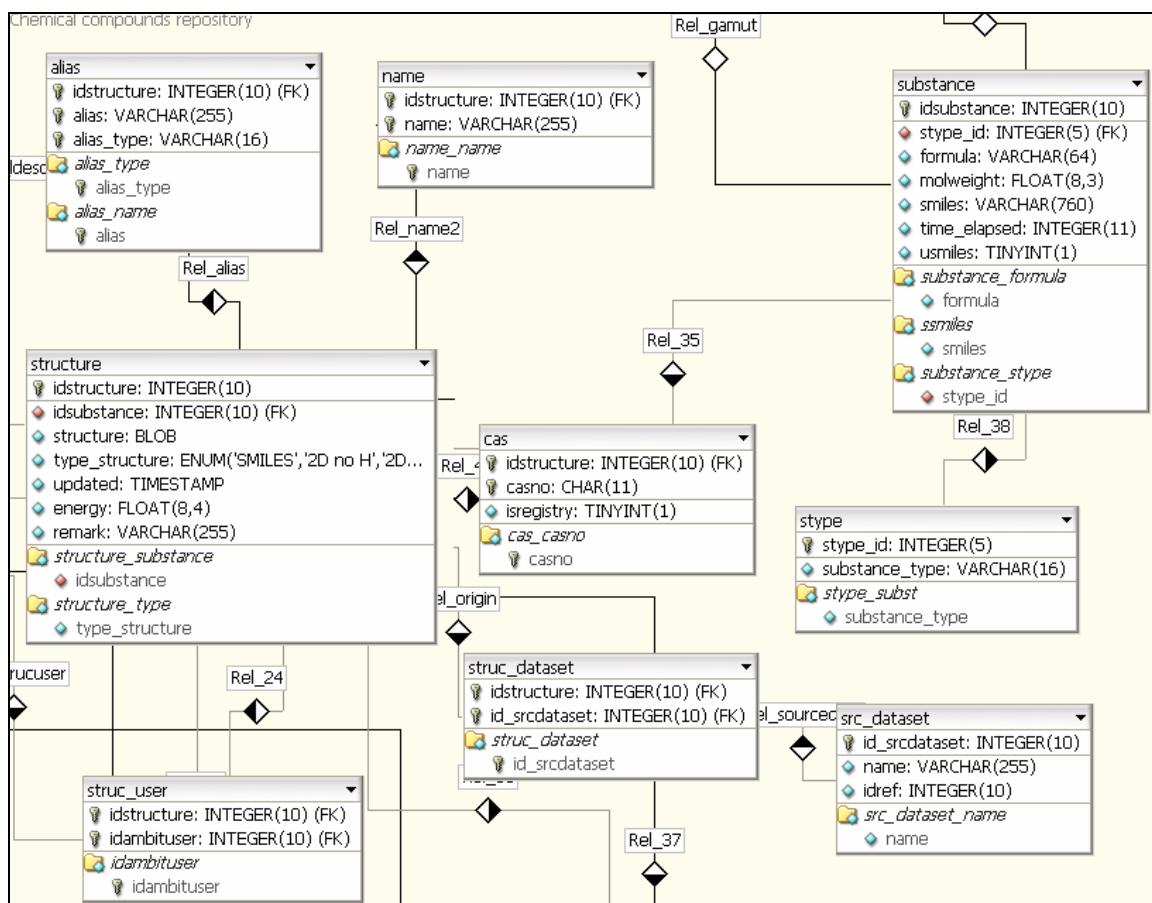


Figure 11. Chemical compounds repository

Table alias
<p>-- An alias. Any string information, different from chemical name and CAS registry number. The alias_type is set by the column alias_type and can be any string of length <= 16.</p> <p>-- This is normally used for registry numbers, different than CAS registry (i.e. NSC number, EINECS, etc.)</p>
<pre>CREATE TABLE alias (idstructure INTEGER(10) UNSIGNED NOT NULL, alias VARCHAR(255) NOT NULL DEFAULT "NA", alias_type VARCHAR(16) NOT NULL DEFAULT "ID", PRIMARY KEY(idstructure, alias, alias_type), INDEX alias_type(alias_type), INDEX alias_name(alias), FOREIGN KEY(idstructure) REFERENCES structure(idstructure) ON DELETE CASCADE ON UPDATE CASCADE)</pre>

Building blocks for QSAR Decision Support System

```
TYPE=InnoDB;
```

Table **cas**

```
-- CAS registry number.  
-- "isregistry" field is here to allow storing of "artificial" casno some  
softwares invented or to track possible errors in CAS numbers.
```

```
CREATE TABLE cas (  
  idstructure INTEGER(10) UNSIGNED NOT NULL,  
  casno CHAR(11) NOT NULL,  
  isregistry TINYINT(1) UNSIGNED NOT NULL DEFAULT "true",  
  PRIMARY KEY(idstructure, casno),  
  INDEX cas_casno(casno),  
  FOREIGN KEY(idstructure)  
    REFERENCES structure(idstructure)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION  
)  
TYPE=InnoDB;
```

Table **name**

```
-- Chemical name. A structure may have any number of names.
```

```
CREATE TABLE name (  
  idstructure INTEGER(10) UNSIGNED NOT NULL,  
  name VARCHAR(255) NOT NULL,  
  PRIMARY KEY(idstructure, name),  
  INDEX name_name(name),  
  FOREIGN KEY(idstructure)  
    REFERENCES structure(idstructure)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE  
)  
TYPE=InnoDB;
```

Table **src_dataset**

```
-- This is to store information of the origin of the compounds. Generally, the  
field "name" stores the name of the file where compounds are read from.  
-- idref is a link to a literature reference (if available)
```

```
CREATE TABLE src_dataset (  
  id_srcdataset INTEGER(10) UNSIGNED NOT NULL AUTO_INCREMENT,  
  name VARCHAR(255) NOT NULL DEFAULT "default",  
  idref INTEGER(10) UNSIGNED NULL,  
  PRIMARY KEY(id_srcdataset),  
  UNIQUE INDEX src_dataset_name(name)  
)  
TYPE=InnoDB;
```

Building blocks for QSAR Decision Support System

Table structure
<pre>-- This is where chemical structures are stored. idsubstance is a link to the table substance. -- The field structure contains the structure itself in a compressed CML format. -- The type_structure has one of the following values : -- SMILES - i.e. the structure was generated from SMILES and there is no 2d and 3d information -- 2D without Hydrogens -- 2D with Hydrogens -- 3D without Hydrogens -- 3D with Hydrogens -- The field remark is a free text comment.</pre>
<pre>CREATE TABLE structure (idstructure INTEGER(10) UNSIGNED NOT NULL AUTO_INCREMENT, idsubstance INTEGER(10) UNSIGNED NOT NULL, structure BLOB NOT NULL, type_structure ENUM('SMILES','2D no H','2D with H','3D no H','3D with H','optimized','experimental') NOT NULL DEFAULT "2D with H", updated TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP, energy FLOAT(8,4) NOT NULL DEFAULT 0, remark VARCHAR(255) NOT NULL, PRIMARY KEY(idstructure), INDEX structure_substance(idsubstance), INDEX structure_type(type_structure), FOREIGN KEY(idsubstance) REFERENCES substance(idsubstance) ON DELETE RESTRICT ON UPDATE CASCADE) TYPE=InnoDB;</pre>

Table struc_dataset
<pre>-- Which structure comes from which dataset (see src_dataset table) ----- CREATE TABLE struc_dataset (idstructure INTEGER(10) UNSIGNED NOT NULL, id_srcdataset INTEGER(10) UNSIGNED NOT NULL, PRIMARY KEY(idstructure, id_srcdataset), INDEX struc_dataset(id_srcdataset), FOREIGN KEY(idstructure) REFERENCES structure(idstructure) ON DELETE CASCADE ON UPDATE CASCADE, FOREIGN KEY(id_srcdataset) REFERENCES src_dataset(id_srcdataset) ON DELETE CASCADE</pre>

Building blocks for QSAR Decision Support System

```
        ON UPDATE CASCADE
    )
TYPE=InnoDB;

-----
-- This is to keep track which user has been entered each particular structure.
-----

CREATE TABLE struc_user (
  idstructure INTEGER(10) UNSIGNED NOT NULL,
  idambituser INTEGER(10) UNSIGNED NOT NULL,
  PRIMARY KEY(idstructure, idambituser),
  INDEX idambituser(idambituser),
  FOREIGN KEY(idstructure)
    REFERENCES structure(idstructure)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
  FOREIGN KEY(idambituser)
    REFERENCES ambituser(idambituser)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
)
TYPE=InnoDB;

-----
-- The substance type, i.e. organic, inorganic, metalloorganic, etc.
-----

CREATE TABLE stype (
  stype_id INTEGER(5) UNSIGNED NOT NULL AUTO_INCREMENT,
  substance_type VARCHAR(16) NOT NULL,
  PRIMARY KEY(stype_id),
  UNIQUE INDEX stype_subst(substance_type)
)
TYPE=InnoDB;
INSERT INTO stype (stype_id, substance_type) VALUES(null, 'organic');
INSERT INTO stype (stype_id, substance_type) VALUES(null, 'inorganic');
INSERT INTO stype (stype_id, substance_type) VALUES(null, 'organometallic');
INSERT INTO stype (stype_id, substance_type) VALUES(null, 'mixture' );
INSERT INTO stype (stype_id, substance_type) VALUES(null, 'unknown' );
-----
-----
```

Table **substance**

```
-- This is the main entry for a chemical compound. idsubstance is automatically
generated unique identifier. stype_id is a link to table stype (substance
type).
-- Formula, molecular weight and smiles are stored in the corresponding fields.
Smiles may be missing.
-- usmiles is a flag where the smiles is a canonical one.
-- The actual structure is stored in the table structure. There may be multiple
structures (i.e. conformers or just with different details , coming from
```

Building blocks for QSAR Decision Support System

```
different origins).  
CREATE TABLE substance (  
  idsubstance INTEGER(10) UNSIGNED NOT NULL AUTO_INCREMENT,  
  stype_id INTEGER(5) UNSIGNED NOT NULL DEFAULT 1,  
  formula VARCHAR(64) BINARY NOT NULL,  
  molweight FLOAT(8,3) NOT NULL,  
  smiles VARCHAR(760) BINARY NULL,  
  time_elapsed INTEGER(11) UNSIGNED NULL,  
  usmiles TINYINT(1) UNSIGNED NULL,  
  PRIMARY KEY(idsubstance),  
  INDEX substance_formula(formula),  
  INDEX ssmiles(smiles(760)),  
  INDEX substance_stype(stype_id),  
  FOREIGN KEY(stype_id)  
    REFERENCES stype(stype_id)  
    ON DELETE RESTRICT  
    ON UPDATE CASCADE  
)  
TYPE=InnoDB;
```

Figure 12. SQL script to create chemical compounds repository

Building blocks for QSAR Decision Support System

2.6.2. Search speedup tables

Chemical compounds repository : Search speed up

gamut	fp1024
↳ idsubstance: INTEGER(10) (FK)	↳ idsubstance: INTEGER(10) (FK)
↳ C: INTEGER(10)	↳ fp1: BIGINT(20)
↳ O: INTEGER(10)	↳ fp2: BIGINT(20)
↳ N: INTEGER(10)	↳ fp3: BIGINT(20)
↳ S: INTEGER(10)	↳ fp4: BIGINT(20)
↳ P: INTEGER(10)	↳ fp5: BIGINT(20)
↳ Cl: INTEGER(10)	↳ fp6: BIGINT(20)
↳ F: INTEGER(10)	↳ fp7: BIGINT(20)
↳ Br: INTEGER(10)	↳ fp8: BIGINT(20)
↳ I: INTEGER(10)	↳ fp9: BIGINT(20)
↳ Si: INTEGER(10)	↳ fp10: BIGINT(20)
↳ B: INTEGER(10)	↳ fp11: BIGINT(20)
↳ aa: INTEGER(10)	↳ fp12: BIGINT(20)
↳ rno: INTEGER(10)	↳ fp13: BIGINT(20)
↳ rmax: INTEGER(10)	↳ fp14: BIGINT(20)
↳ b1: INTEGER(10)	↳ fp15: BIGINT(20)
↳ bar: INTEGER(10)	↳ fp16: BIGINT(20)
↳ b2: INTEGER(10)	↳ time: INTEGER(10)
↳ b3: INTEGER(10)	↳ bc: INTEGER(6)
↳ ab: INTEGER(10)	↳ <i>fpall</i>
↳ cb: INTEGER(10)	↳ fp1
↳ st: INTEGER(10)	↳ fp2
↳ o1: INTEGER(10)	↳ fp3
↳ o2: INTEGER(10)	↳ fp4
↳ o3: INTEGER(10)	↳ fp5
↳ o4: INTEGER(10)	↳ fp6
↳ oo: INTEGER(10)	↳ fp7
↳ <i>gamut_all</i>	↳ fp8
↳ aa	↳ fp9
↳ C	↳ fp10
↳ O	↳ fp11
↳ N	↳ fp12
↳ S	↳ fp13
↳ P	↳ fp14
↳ Cl	↳ fp15
↳ F	↳ fp16
↳ Br	

Figure 13. Search speedup tables

Building blocks for QSAR Decision Support System

```
-- Stores 1024 length fingerprints
-- -----
CREATE TABLE fp1024 (
  idsubstance INTEGER(10) UNSIGNED NOT NULL,
  fp1 BIGINT(20) UNSIGNED NOT NULL,
  fp2 BIGINT(20) UNSIGNED NOT NULL,
  fp3 BIGINT(20) UNSIGNED NOT NULL,
  fp4 BIGINT(20) UNSIGNED NOT NULL,
  fp5 BIGINT(20) UNSIGNED NOT NULL,
  fp6 BIGINT(20) UNSIGNED NOT NULL,
  fp7 BIGINT(20) UNSIGNED NOT NULL,
  fp8 BIGINT(20) UNSIGNED NOT NULL,
  fp9 BIGINT(20) UNSIGNED NOT NULL,
  fp10 BIGINT(20) UNSIGNED NOT NULL,
  fp11 BIGINT(20) UNSIGNED NOT NULL,
  fp12 BIGINT(20) UNSIGNED NOT NULL,
  fp13 BIGINT(20) UNSIGNED NOT NULL,
  fp14 BIGINT(20) UNSIGNED NOT NULL,
  fp15 BIGINT(20) UNSIGNED NOT NULL,
  fp16 BIGINT(20) UNSIGNED NOT NULL,
  time INTEGER(10) UNSIGNED NOT NULL,
  bc INTEGER(6) UNSIGNED NOT NULL,
  PRIMARY KEY(idsubstance),
  INDEX fpall(fp1, fp2, fp3, fp4, fp5, fp6, fp7, fp8, fp9, fp10, fp11, fp12,
fp13, fp14, fp15, fp16),
  FOREIGN KEY(idsubstance)
    REFERENCES substance(idsubstance)
    ON DELETE CASCADE
    ON UPDATE CASCADE
)
TYPE=InnoDB;
-- -----
-- Atom counts , bond counts and other useful information
-- -----
CREATE TABLE gamut (
  idsubstance INTEGER(10) UNSIGNED NOT NULL,
```

Building blocks for QSAR Decision Support System

```
C INTEGER(10) UNSIGNED NOT NULL,  
O INTEGER(10) UNSIGNED NOT NULL,  
N INTEGER(10) UNSIGNED NOT NULL,  
S INTEGER(10) UNSIGNED NOT NULL,  
P INTEGER(10) UNSIGNED NOT NULL,  
Cl INTEGER(10) UNSIGNED NOT NULL,  
F INTEGER(10) UNSIGNED NOT NULL,  
Br INTEGER(10) UNSIGNED NOT NULL,  
I INTEGER(10) UNSIGNED NOT NULL,  
Si INTEGER(10) UNSIGNED NOT NULL,  
B INTEGER(10) UNSIGNED NOT NULL,  
aa INTEGER(10) UNSIGNED NOT NULL,  
rno INTEGER(10) UNSIGNED NOT NULL,  
rmax INTEGER(10) UNSIGNED NOT NULL,  
b1 INTEGER(10) UNSIGNED NOT NULL,  
bar INTEGER(10) UNSIGNED NOT NULL,  
b2 INTEGER(10) UNSIGNED NOT NULL,  
b3 INTEGER(10) UNSIGNED NOT NULL,  
ab INTEGER(10) UNSIGNED NOT NULL,  
cb INTEGER(10) UNSIGNED NOT NULL,  
st INTEGER(10) UNSIGNED NOT NULL,  
o1 INTEGER(10) UNSIGNED NOT NULL,  
o2 INTEGER(10) UNSIGNED NOT NULL,  
o3 INTEGER(10) UNSIGNED NOT NULL,  
o4 INTEGER(10) UNSIGNED NOT NULL,  
oo INTEGER(10) UNSIGNED NOT NULL,  
PRIMARY KEY(idsubstance),  
INDEX gamut_all(aa, C, O, N, S, P, Cl, F, Br, I, Si, B),  
INDEX gamut_bonds(ab, cb, b1, bar, b2, b3),  
INDEX gamut_o(o1, o2, o3, o4, oo),  
FOREIGN KEY(idsubstance)  
REFERENCES substance(idsubstance)  
ON DELETE CASCADE  
ON UPDATE CASCADE  
)  
TYPE=InnoDB;
```

Building blocks for QSAR Decision Support System

Figure 14. SQL script to create search speedup tables

2.6.3. Query tables

These are to hold subsets, selected by an user.

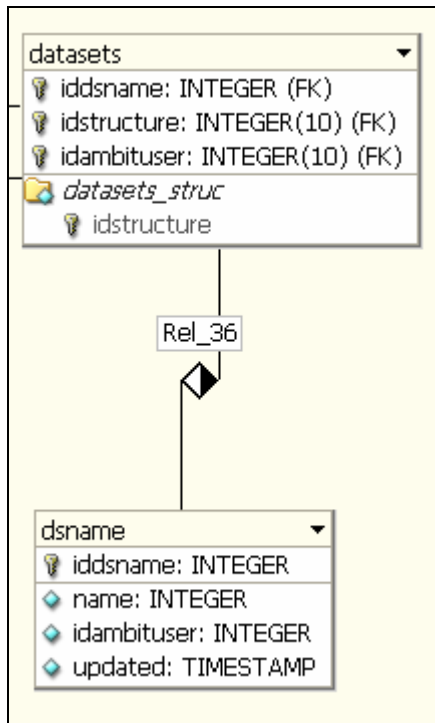


Figure 15. Query tables

```
CREATE TABLE datasets (  
  iddsname INTEGER UNSIGNED NOT NULL,  
  idstructure INTEGER(10) UNSIGNED NOT NULL,  
  idambituser INTEGER(10) UNSIGNED NOT NULL,  
  PRIMARY KEY(iddsname, idstructure, idambituser),  
  INDEX datasets_struc(idstructure),  
  FOREIGN KEY(iddsname)  
    REFERENCES dsname(iddsname)
```

Building blocks for QSAR Decision Support System

```
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
FOREIGN KEY(idstructure)
REFERENCES structure(idstructure)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
FOREIGN KEY(idambituser)
REFERENCES ambituser(idambituser)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
)
TYPE=InnoDB;

CREATE TABLE dsname (
    iddsname INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
    name INTEGER UNSIGNED NOT NULL,
    idambituser INTEGER UNSIGNED NOT NULL,
    updated TIMESTAMP NOT NULL,
    PRIMARY KEY(iddsname)
)
TYPE=InnoDB;
```

Figure 16. SQL script to create query tables

3.Database API (Application programming interface)

All the applications within the AMBIT project are implemented in Java.

The basic cheminformatics functionality relies on the open source Java library - [The Chemistry Development Kit, CDK](#).

The source code is organized in modules (packages). The list of packages is shown below. The packages implementing the Database API are shown in bold. For complete documentation see <http://luna.acad.bg/nina/projects/doc/api/index.html>

	Packages	Description
--	-----------------	--------------------

Building blocks for QSAR Decision Support System

1	ambit.applications	GUI and console applications for the AMBIT project.
2	ambit.benchmark	Some simple benchmarking facilities for the AMBIT project.
3	ambit.data	Implementation of the basic data structures for the AMBIT project.
4	ambit.data.descriptors	Classes to store a descriptor definition
5	ambit.data.domain	Classes to store QSAR data, estimate and assess applicability domain of a QSAR model
6	ambit.data.literature	Classes to store a literature reference
7	ambit.data.model	Classes to store a QSAR model
8	ambit.data.molecule	Implementation of a chemical compound functionality
9	ambit.data.studies	Classes for species, studies, experimental results
10	ambit.database	Database API for the MySQL AMBIT database
11	ambit.database.actions	Database API for the MySQL AMBIT database.
12	ambit.database.acquire	Database API for the MySQL AMBIT database.
13	ambit.database.compounds	Database API for the MySQL AMBIT database.
14	ambit.datastructures	Data structures to analyze a chemical compound
15	ambit.io	Implementation of text file input/output for the AMBIT project.
16	ambit.log	Logging functionality
17	ambit.misc	
18	ambit.stats	Statistics basic classes
19	ambit.stats.datastructures	Datastructures for the statistics classes
20	ambit.stats.transforms	Statistics : Transformations Fast Fourie Transform
21	ambit.stats.transforms.densityestimation	Statistics : Density estimation Non parametric kernel density estimation
22	ambit.stats.transforms.transformfilters	Data transformations (PCA and more)
23	ambit.test	JUnit tests

Building blocks for QSAR Decision Support System

24	ambit.test.database	JUnit tests for the Database API
25	ambit.ui	User Interface core classes
26	ambit.ui.data	User Interface classes for the AmbitObject and AmbitList descendants
27	ambit.ui.domain	User Interface classes for the ambit.data.domain classes
28	ambit.ui.query	User Interface classess for the database queries

4. User interface

There are two console application `AmbitImportConsole` to import compounds from SDF file into database and `AmbitGenerateConsole` to generate smiles and fingerprints. Since these are batch tasks, this functionality has no graphical user interface at this moment.

`AmbitDB` is an GUI application to enter, modify data and for database administration was created. It allows to connect to the database, manage users, edit and insert all data types and perform administrative tasks as dropping and creating tables. Since it is in its early stage, only a brief description will follow.

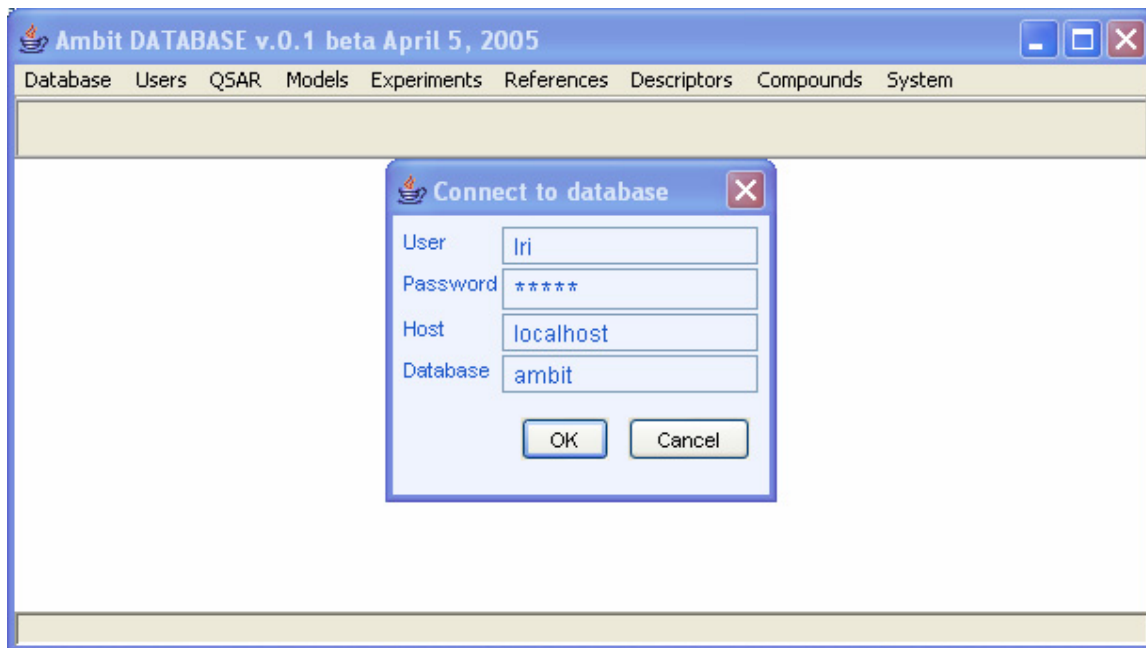
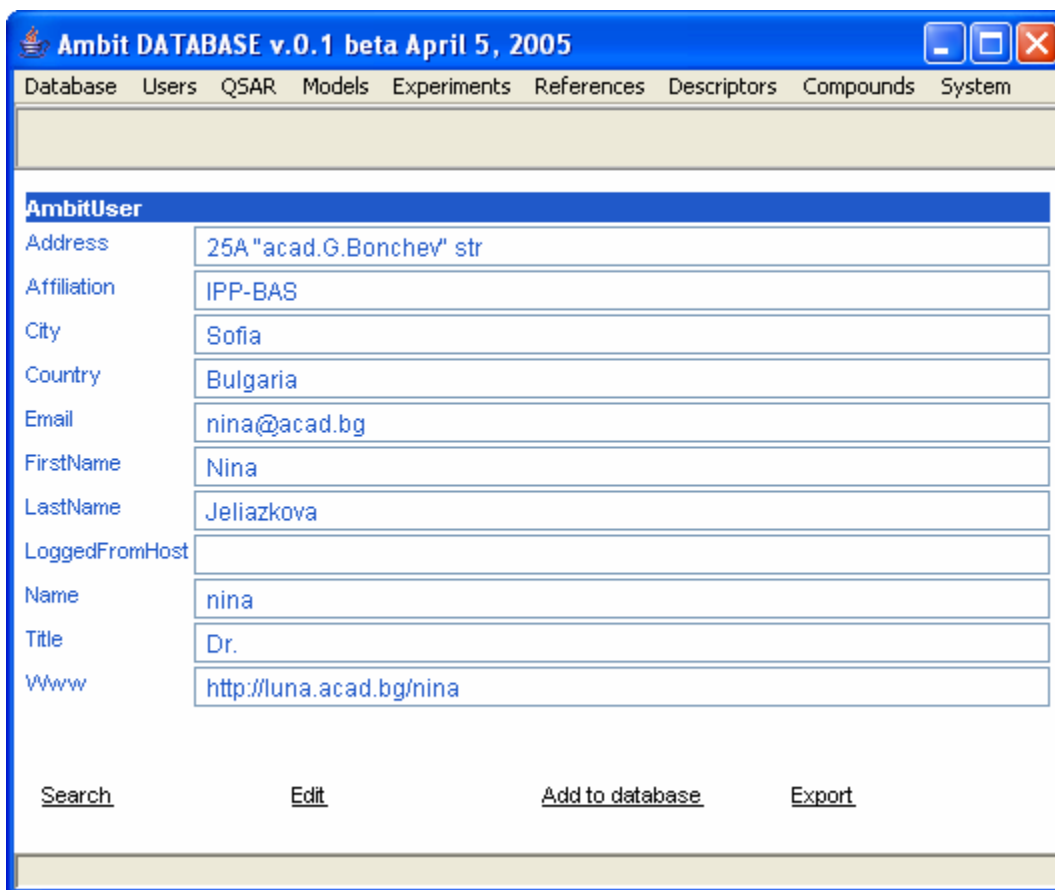


Figure 17. Connecting to database

Building blocks for QSAR Decision Support System



The screenshot shows a software window titled "Ambit DATABASE v.0.1 beta April 5, 2005". The window has a menu bar with the following items: Database, Users, QSAR, Models, Experiments, References, Descriptors, Compounds, and System. Below the menu bar is a large empty rectangular area. Underneath that is a blue header bar labeled "AmbitUser". Below the header is a form with the following fields and values:

Address	25A "acad.G.Bonchev" str
Affiliation	IPP-BAS
City	Sofia
Country	Bulgaria
Email	nina@acad.bg
FirstName	Nina
LastName	Jeliazkova
LoggedFromHost	
Name	nina
Title	Dr.
Www	http://luna.acad.bg/nina

At the bottom of the form are four buttons: Search, Edit, Add to database, and Export.

Figure 18. Entering a new user

Building blocks for QSAR Decision Support System

The screenshot shows a software window titled "Ambit DATABASE v.0.1 beta April 5, 2005". The window has a menu bar with the following items: Database, Users, QSAR, Models, Experiments, References, Descriptors, Compounds, and System. Below the menu bar is a large empty rectangular area. At the bottom of the window, there is a section titled "QSARDataset" with several fields and buttons:

QSARDataset	
Coverage	No method assigned not assessed
Data	Data: 608 points 1 descriptor(s)
Model	Model,"BCFWIN model","Linear Regression", ""
Name	BCFWIN model
Ndescriptors	1
Npoints	608
Yname	Estimated Log BCF =

Below the table, there are four buttons: [Search](#), [Edit](#), [Add to database](#), and [Export](#).

Figure 19. a QSAR Data set editing

Clicking on the Model box allows to edit the Model

Building blocks for QSAR Decision Support System

Ambit DATABASE v.0.1 beta April 5, 2005

Database Users QSAR Models Experiments References Descriptors Compounds System

[Back to Model,"BCFWIN model","Linear Reg](#)

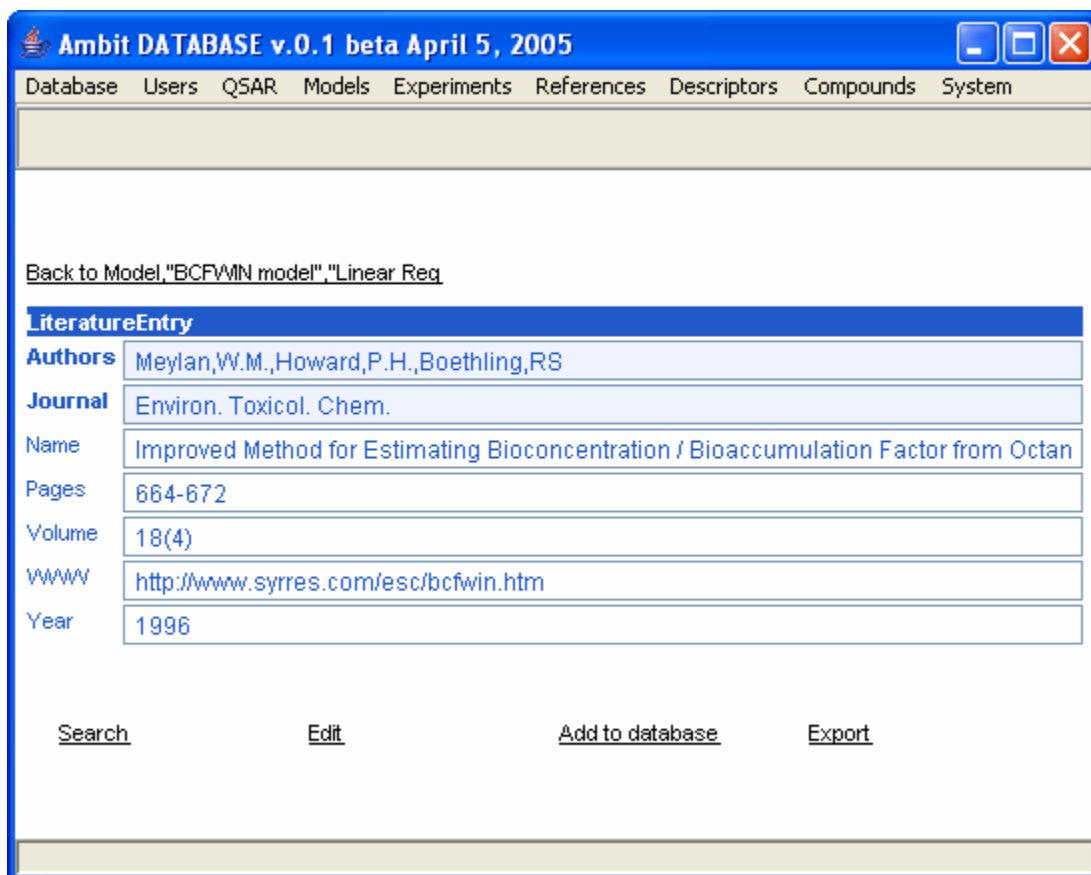
Model	
Descriptors	0.log_P
Equation	
FileWithData	
Keywords	Bioconcentration factor; QSAR
ModelType	Linear Regression
N_Descriptors	1
N_Points	0
Name	BCFWIN model
Note	
R2	0.0
Reference	Meylan,W.M.,Howard,P.H.,Boethling,RS,"Improved Method for Estimating Bioconce
Study	...

[Search](#) [Edit](#) [Add to database](#) [Export](#)

Figure 20. a Model editing

Clicking on the reference box allows to edit the reference

Building blocks for QSAR Decision Support System



The screenshot shows a software window titled "Ambit DATABASE v.0.1 beta April 5, 2005". The window has a menu bar with the following items: Database, Users, QSAR, Models, Experiments, References, Descriptors, Compounds, and System. Below the menu bar is a large empty text area. Underneath this area is a link: [Back to Model,"BCFWIN model","Linear Reg](#). Below the link is a section header "LiteratureEntry" in a blue bar. This section contains several input fields for a literature entry:

Authors	Meylan,W.M.,Howard,P.H.,Boethling,RS
Journal	Environ. Toxicol. Chem.
Name	Improved Method for Estimating Bioconcentration / Bioaccumulation Factor from Octan
Pages	664-672
Volume	18(4)
WWW	http://www.syrres.com/esc/bcfwin.htm
Year	1996

At the bottom of the form are four buttons: [Search](#), [Edit](#), [Add to database](#), and [Export](#).

Figure 21. a reference editing

Clicking on Authors box allows to edit the list of the authors

Building blocks for QSAR Decision Support System

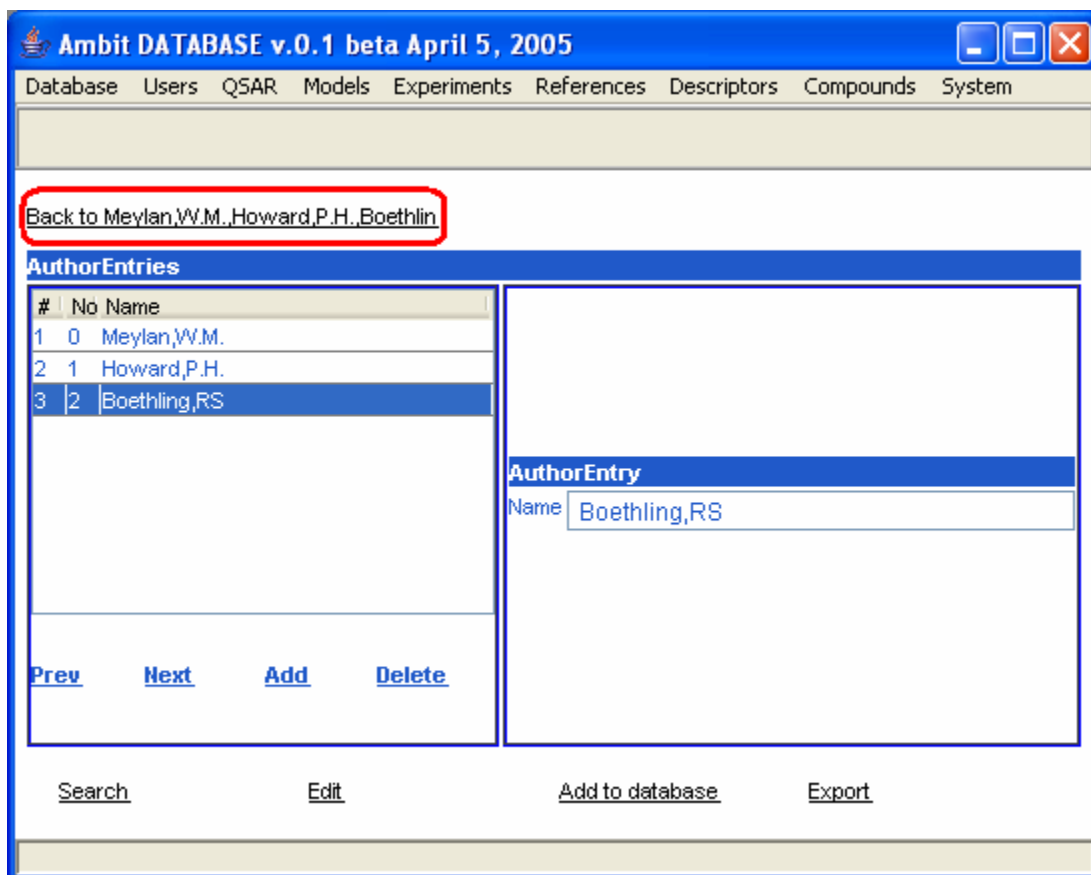
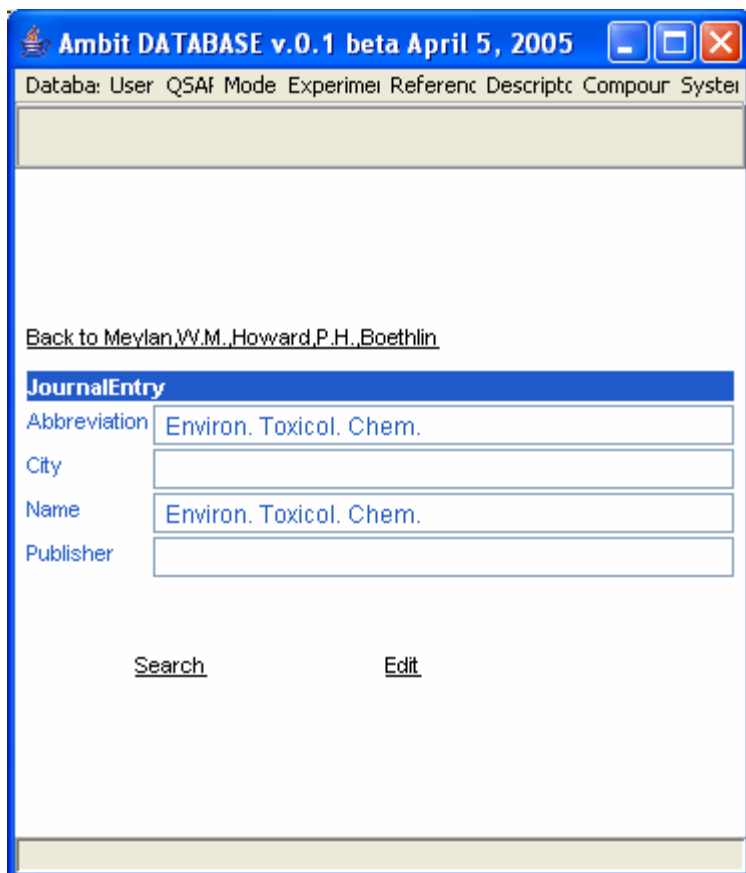


Figure 22. Authors list editing

A click on the red-marked link leads back to the screen of editing (Figure 21). There, we can click on the Journal box and edit the journal entry

Building blocks for QSAR Decision Support System



The screenshot shows a web browser window titled "Ambit DATABASE v.0.1 beta April 5, 2005". The address bar contains "Databa: User QSAF Mode Experime Referenc Descriptc Compour System". Below the address bar is a navigation menu with the following items: "Back to Meylan,W.M.,Howard,P.H.,Boethlin". The main content area features a blue header "JournalEntry" followed by four input fields: "Abbreviation" (containing "Environ. Toxicol. Chem."), "City" (empty), "Name" (containing "Environ. Toxicol. Chem."), and "Publisher" (empty). At the bottom of the form are two buttons: "Search" and "Edit".

Figure 23. Authors list editing

Several clicks on the **“Back to ...”** link follows to the initial screen (Figure 19). Any non-white box means that it has more complex information than a single string or number and can be clicked to follow to its own screen.

Search, **Add To Database** and **Export** provides a common way to perform these functions for different data. On clicking on the **Search** link a dialog appears, where search criteria can be entered. Then the results (if any) appear as a list like the lists at these last figures.

For example, a list of experimental results may look like as follows:

Building blocks for QSAR Decision Support System

The screenshot shows the Ambit DATABASE v.0.1 beta April 5, 2005 interface. The main window has a menu bar with options: Database, Users, QSAR, Models, Experiments, References, Descriptors, Compounds, and System. Below the menu bar, there is a navigation link: [Back to Model,"BCFWMN model","Linear Reg](#).

The main content area is divided into two sections. On the left is the **ExperimentList** table, and on the right is the **Experiment** details form.

#	No	Name
1	0	Ames ,log rev/hmol,,;Salmonella typhim...
2	1	Ames ,log rev/hmol,,;Salmonella typhim...
3	2	Ames ,log rev/hmol,,;Salmonella typhim...
4	3	Ames ,log rev/hmol,,;Salmonella typhim...
5	4	Ames ,log rev/hmol,,;Salmonella typhim...
6	5	Ames ,log rev/hmol,,;Salmonella typhim...

Below the table are navigation buttons: [Prev](#), [Next](#), [Add](#), and [Delete](#).

The **Experiment** details form contains the following fields:

- Exp_error**: 0.0
- Mol**: 92-67-1,"4-Aminobiphenyl",Nc1ccc(cc1)c
- Name**: (empty)
- Note**: (empty)
- Reference**: Glende C.,Schmitt H.,Erdinger L.,Boche C
- Result**: 0.65
- Result_str**: (empty)
- Study**: Ames ,log rev/hmol,,;Salmonella typhimur

At the bottom of the interface, there are four main action buttons: [Search](#), [Edit](#), [Add to database](#), and [Export](#).

Figure 24. A demo of experimental results list

Clicking on the Molecule box leads to

Building blocks for QSAR Decision Support System

Ambit DATABASE v.0.1 beta April 5, 2005

Database Users QSAR Models Experiments References Descriptors Compounds System

Back to Model, "BCFMIN model", "Linear Reg

ExperimentList

#	No	Name
1	0	Ames_log rev/hmol,,,Salmonella typhimuri...
2	1	Ames_log rev/hmol,,,Salmonella typhimuri...
3	2	Ames_log rev/hmol,,,Salmonella typhimuri...
4	3	Ames_log rev/hmol,,,Salmonella typhimuri...
5	4	Ames_log rev/hmol,,,Salmonella typhimuri...
6	5	Ames_log rev/hmol,,,Salmonella typhimuri...

Back to Ames_log rev/hmol,,,Salmonella t

Molecule

CAS 153-78-6

Formula

Name 2-AMINOFUORENE

SMILES Nc2ccc3c1ccccc1Cc3(c2)

Structure diagram

Prev Next Add Delete

Search Edit Add to database Export

Figure 25. A demo of experimental results list with 2d representation of a chemical structure
A new compound can be added by clicking on the **Add** link just below the list at the left. Smiles can be edited and the corresponding structure drawing appears. Entering the structure by drawing the 2D picture will be available in the next releases.

Building blocks for QSAR Decision Support System

The screenshot displays the Ambit DATABASE v.0.1 beta April 5, 2005 interface. The main window features a menu bar (Database, Users, QSAR, Models, Experiments, References, Descriptors, Compounds, System) and a table titled 'AmbitList' with columns '#', 'No', and 'Name'. The table contains four entries, with the fourth entry selected: '# 4', 'No 3', 'Name 91-59-8, "2-Aminonaphthalene", Nc1ccc...'. To the right of the table is a 'Molecule' details panel with fields for CAS (91-59-8), Formula, Name (2-Aminonaphthalene), and SMILES (Nc1ccc2ccccc2(c1)). Below these fields is a 'Structure diagram' showing the chemical structure of 2-aminonaphthalene. In the foreground, a 'New item' dialog box is open, containing a 'Molecule' section with fields for CAS, Formula, Name, and SMILES (cccccccc). Below these fields is a 'Structure diagram' showing a zigzag line representing a long alkane chain. The dialog box has 'OK' and 'Cancel' buttons. The main window also has 'Add to database' and 'Export' buttons at the bottom right.

Figure 26. A list of compounds with a new compound added

5.Relation to other EEM9 modules:

EEM9- 1a Applicability domain

Ambit database will be used to store structures, descriptors, endpoints and QSAR models. Thus it will make possible to create a storage for QSAR models with descriptor and structure information, which could be used in order to select models and assess applicability domain.

A possible usage scenario is :

Building blocks for QSAR Decision Support System

- ◆ The user is provided an option to search for a QSAR model – by endpoint, reference or some other criteria.
- ◆ The query returns a list of QSAR models, each of which can be browsed and the user select one or more of them to his further analysis.
- ◆ The models can be compared or applicability domain assessed.
- ◆ The user may ask to verify if a list of compounds are suitable to predict by these models.

This functionality is planned in later phases of the project.

EEM9-1b Chemical grouping

Since fingerprints are calculated once upon structure is inserted into the database and stored into a table, structural similarity by fingerprints is quite easy.

For the descriptor based similarity, the system will rely on the imported descriptors and will provide algorithms to assess similarity and descriptor space.

This functionality is planned in later phases of the project.

EEM 9-2 Conversion CAS to SMILES combined with MOA

CAS registry number (CAS RN) and SMILES (Simplified Molecular Input Line Entry Specification) are widely accepted and used as unique molecular identifiers. However, CAS RN is a registry number and has no chemical meaning. Thus, conversion between CAS RN and SMILES is possible only if a predefined list of correspondence between CAS RN and SMILES (or chemical structures, represented in a computer readable format, from which to generate SMILES) .

The CAS-SMILES and SMILES-CAS conversion may already be tested at <http://luna.acad.bg/nina/projects/> , the link **Query Ambit Database**. The CAS number

Building blocks for QSAR Decision Support System

and SMILES are searched within 463426 chemical compounds available in the Ambit database. The relational database return the results in a fraction of a second.

All the SMILES in the database are generated to be canonical, even if SMILES read from the original file were not.

1 .Introduction.....	2
1.1 .Background.....	2
1.2 .Codename origin.....	2
2 .Database schema.....	3
2.1 .User repository.....	4
2.2 .Literature references repository.....	5
2.3 .Experimental results repository.....	7
2.4 .QSAR models repository.....	11
2.5 .Descriptors repository.....	14
2.6 .Chemical compounds repository.....	18
2.6.1 .Chemical structures repository.....	18
2.6.2 .Search speedup tables.....	24
2.6.3 .Query tables.....	27
3 .Database API (Application programming interface).....	28
Description.....	28
GUI and console applications for the AMBIT project.....	29
Some simple benchmarking facilities for the AMBIT project.....	29
Implementation of the basic data structures for the AMBIT project.....	29

Building blocks for QSAR Decision Support System

Classes to store a descriptor definition	29
Classes to store QSAR data, estimate and assess applicability domain of a QSAR model.....	29
Classes to store a literature reference.....	29
Classes to store a QSAR model	29
Implementation of a chemical compound functionality	29
Classes for species, studies, experimental results.....	29
Database API for the mySQL AMBIT database.....	29
Database API for the mySQL AMBIT database.....	29
Database API for the mySQL AMBIT database.....	29
Database API for the mySQL AMBIT database.....	29
Data structures to analyze a chemical compound.....	29
Implementation of text file input/output for the AMBIT project.	29
Logging functionality.....	29
Statistics basic classes.....	29
Datastructures for the statistics classes.....	29
Statistics : Transformations Fast Fourie Transform.....	29
Statistics : Density estimation Non parametric kernel density estimation.....	29
Data transformations (PCA and more)	29
JUnit tests.....	29
JUnit tests for the Database API.....	30
User Interface core classes.....	30

Building blocks for QSAR Decision Support System

User Interface classess for the AmbitObject and AmbitList descendants	30
User Interface classess for the ambit.data.domain classes.....	30
User Interface classess for the database queries	30
4 .User interface	30
5 .Relation to other EEM9 modules:.....	39
EEM9- 1a Applicability domain.....	39
EEM9-1b Chemical grouping.....	40
EEM 9-2 Conversion CAS to SMILES combined with MOA.....	40