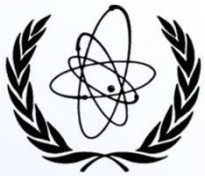
Overview of IT technologies used in X4Pro

Viktor Zerkin
International Atomic Energy Agency, Nuclear Data Section



EXFOR Workshop on the Compilation of Experimental Nuclear Reaction Data
13-16 December 2022, IAEA Headquarters, Vienna, Austria

SQL

Structured Query Language

What is SQL?

SQL (Structured Query Language) is a declarative programming language used to create, modify and manage data in a relational database managed by a database management system.
SQL consists of a collection of operators, statements and calculated functions.

Types of SQL statements:

- *Data Definition Language (DDL):*
  - *CREATE creates a database object (database, table, view, user, and so on),*
  - *ALTER changes the object,*
  - *DROP deletes the object;*
- *Data Manipulation Language (DML):*
  - *SELECT selects data that meets the specified conditions*
  - *INSERT adds new data,*
  - *UPDATE changes existing data,*
  - *DELETE deletes data;*
- *Data Control Language (DCL):*
  - *GRANT grants the user (group) permissions for certain operations with the object,*
  - *REVOKE revokes previously issued permits,*
  - *DENY sets a ban that has priority over resolution;*
- *Transaction Control Language (TCL):*
  - *COMMIT applies the transaction,*
  - *ROLLBACK rolls back all changes made in the context of the current transaction,*
  - *SAVEPOINT divides the transaction into smaller sections.*

SQL examples

Hello, World!

```
select 'Hello, world'
```

How to run it

```
$ sqlite3 x4sqlite1.db "select 'Hello, world',5*3.5-2"
Hello, world|15.5
```

How to execute SQL command from a text file and write result to another file

```
$ sqlite3.exe x4sqlite1.db <sql1.txt >result1.txt
```

Create table using existing tables

```
create table AAA as select * from AUTHORS where Author like 'A%';
select * from AAA;
10017|0|1969|3|10017|R.O.|Akyuz|R.O. Akyuz
10126|0|1970|2|10126|G.P.|Arnold|G.P. Arnold
10141|0|1971|1|10141|B.J.|Allen|B.J. Allen
```

Delete existing table

```
drop table AAA
```

Optimize disk space (after modifications only: drop table, update, delete)

```
vacuum -- SQLite compress database (free unused disk space) very slow
optimize table ENTRY - MySQL, MariaDB
```

SQLite Download Page:
https://www.sqlite.org/download.html

DB Browser for SQLite.
Download: https://sqlitebrowser.org/dl/

Install using command lines.
Linux/Ubuntu:
$ sudo apt-get install sqlite3
$ sudo apt-get install sqlitebrowser

Our main interest: SELECT command

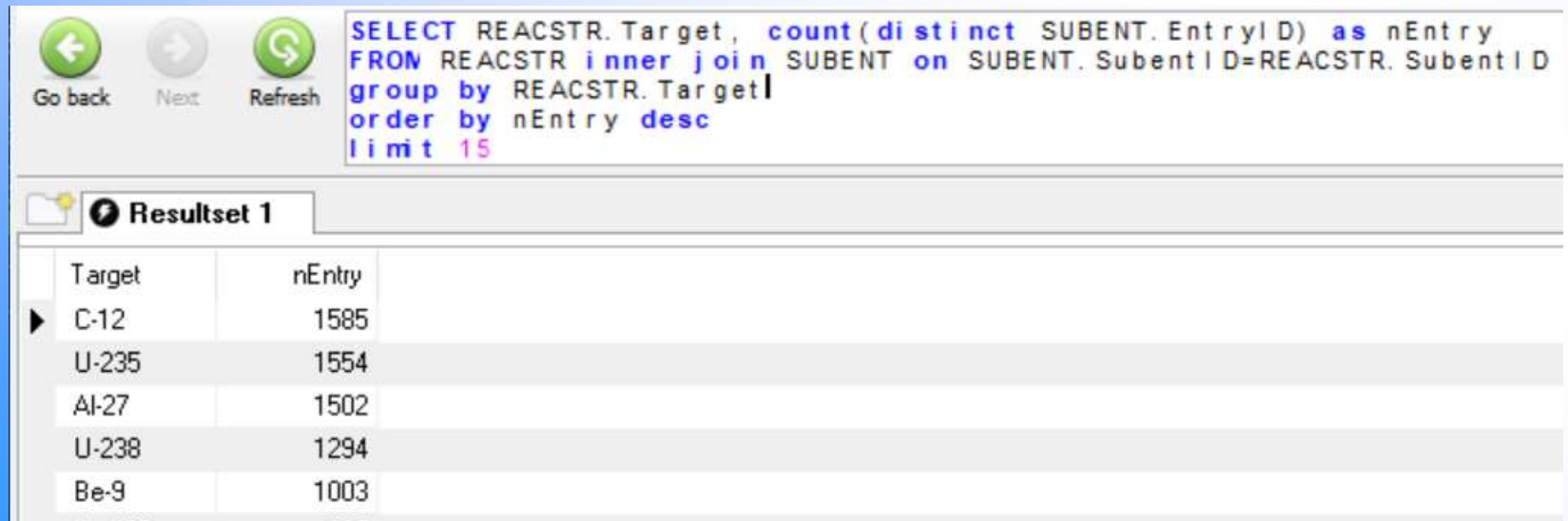
```
SELECT 'Example 1',Entry,DateDebut,Author1ini,Author1,TransDate,TransFile,Reference1,nAuthors
FROM  ENTRY
WHERE Author1='Korzh' AND YearRef1>=1977 AND nAuthors<5
ORDER BY Entry
LIMIT 1,4
Example 1|32230|2009-12-11|I.A.|Korzh|20100126|EXFOR-2015-05-05.bck||J,AE,62,(6),417,1987|3
Example 1|40618|1983-03-29|I.A.|Korzh|20210914|trans.4197||J,UFZ,25,(1),109,1980|3
Example 1|40619|1982-11-17|I.A.|Korzh|20180525|trans.4178||J,YF,35,1097,1982|4
Example 1|40858|1985-05-05|I.A.|Korzh|20050926|EXFOR-2015-05-05.bck||C,83KIEV,3,167,198310|3
```

SELECT statement

```
SELECT
[ALL | DISTINCT]
select_expr [, select_expr] ...
[FROM table_references ]
[WHERE where_condition]
[GROUP BY {col_name | expr | position}, ... ]
[HAVING where_condition]
[ORDER BY {col_name | expr | position} [ASC | DESC], ...]
[LIMIT {[offset,] row_count] }
```

```
SELECT * FROM x4pro_ds                                          -- all columns, all rows
SELECT DatasetID FROM x4pro_ds                                  -- one column, all rows
SELECT SUBSTRING(DatasetID,1,5) AS Entry FROM x4pro_ds ORDER BY Entry DESC  -- Entry, all rows
SELECT DISTINCT SUBSTRING(DatasetID,1,5) AS Entry FROM x4pro_ds             -- Entry, unique rows
SELECT COUNT(DISTINCT SUBSTRING(DatasetID,1,5)) AS nEntry FROM x4pro_ds     -- count unique Entries
```

```
SELECT REACSTR.Target, count(distinct SUBENT.EntryID) as nEntry
FROM REACSTR inner join SUBENT on SUBENT.SubentID=REACSTR.SubentID
group by REACSTR.Target
order by nEntry desc
limit 15
```



Target	nEntry
C-12	1585
U-235	1554
Al-27	1502
U-238	1294
Be-9	1003

SELECT statement. EXFOR statistics: Targets

```
-- EXFOR contents statistics: Targets
select date() as Today,'EXFOR' as DB  -- MariaDB: date(now())
,(select max(date(UpdateDate)) from X4UPDATE) as Version
,(select count(EntryID) as nEntries from ENTRY) as nEntryAll
,(select count(distinct EntryID) from SUBENT where DnRow>0) as nEntryDat
,REACSTR.Target,count(distinct SUBENT.EntryID) as nEntry
,round(count(distinct SUBENT.EntryID)*100./(select count(EntryID) from ENTRY),2) as Percent
from REACSTR
inner join SUBENT on SUBENT.SubentID=REACSTR.SubentID
where REACSTR.Target like '%2%'
group by REACSTR.Target
having nEntry>700 and Percent>=3
order by nEntry desc
limit 4
```

```
1   -- EXFOR contents statistics: Targets
2   select date() as Today,'EXFOR' as DB
3   ,(select max(date(UpdateDate)) from X4UPDATE) as Version
4   ,(select count(EntryID) as nEntries from ENTRY) as nEntryAll
5   ,(select count(distinct EntryID) from SUBENT where DnRow>0) as nEntryDat
6   ,REACSTR.Target,count(distinct SUBENT.EntryID) as nEntry
7   ,round(count(distinct SUBENT.EntryID)*100./(select count(EntryID) from ENTRY),2) as Percent
8   from REACSTR
9   inner join SUBENT on SUBENT.SubentID=REACSTR.SubentID
10  where REACSTR.Target like '%2%'
11  group by REACSTR.Target
12  having nEntry>700 and Percent>=3
13  order by nEntry desc
14  limit 4
```

	Today	DB	Version	nEntryAll	nEntryDat	Target	nEntry	Percent
1	2022-12-11	EXFOR	2022-11-28	26131	24830	C-12	1585	6.07
2	2022-12-11	EXFOR	2022-11-28	26131	24830	U-235	1554	5.95
3	2022-12-11	EXFOR	2022-11-28	26131	24830	Al-27	1502	5.75
4	2022-12-11	EXFOR	2022-11-28	26131	24830	U-238	1294	4.95

```
Execution finished without errors.
Result: 4 rows returned in 974ms
```

#Entries grouped by Targets for γ-induced reactions

```
-- #Entries group by Targets where Projectile='g'
select date() as Today,'EXFOR' as DB
,(select max(date(UpdateDate)) from X4UPDATE) as Version
,(select count(EntryID) as nEntries from ENTRY) as nEntryAll
,(select count(distinct EntryID) from SUBENT where DnRow>0) as nEntryDat
,(select count(distinct EntryID) from SUBENT inner join REACSTR on SUBENT.SubentID=REACSTR.SubentID
  where DnRow>0 and REACSTR.Projectile like 'g') as nEntryGamma
,REACSTR.Target,count(distinct SUBENT.EntryID) as nEntry
,round(count(distinct SUBENT.EntryID)*100./
(select count(distinct EntryID) from SUBENT inner join REACSTR on SUBENT.SubentID=REACSTR.SubentID
 where DnRow>0 and REACSTR.Projectile like 'g')
,2) as Percent
from REACSTR inner join SUBENT on SUBENT.SubentID=REACSTR.SubentID
where REACSTR.Projectile like 'g'
group by REACSTR.Target
having nEntry>10 order by nEntry desc limit 10
```

```
1  -- #Entries group by Targets where Projectile='g'
2  select date() as Today,'EXFOR' as DB
3  ,(select max(date(UpdateDate)) from X4UPDATE) as Version
4  ,(select count(EntryID) as nEntries from ENTRY) as nEntryAll
5  ,(select count(distinct EntryID) from SUBENT where DnRow>0) as nEntryDat
6  ,(select count(distinct EntryID) from SUBENT inner join REACSTR on SUBENT.SubentID=REACSTR.SubentID
7    where DnRow>0 and REACSTR.Projectile like 'g') as nEntryGamma
8  ,REACSTR.Target,count(distinct SUBENT.EntryID) as nEntry
9  ,round(count(distinct SUBENT.EntryID)*100./
10 (select count(distinct EntryID) from SUBENT inner join REACSTR on SUBENT.SubentID=REACSTR.SubentID
11  where DnRow>0 and REACSTR.Projectile like 'g')
12 ,2) as Percent
13 from REACSTR
14 inner join SUBENT on SUBENT.SubentID=REACSTR.SubentID
15 where REACSTR.Projectile like 'g'
16 group by REACSTR.Target
17 having nEntry>10
18 order by nEntry desc
19 limit 10
20
```

	Today	DB	Version	nEntryAll	nEntryDat	nEntryGamma	Target	nEntry	Percent
1	2022-12-12	EXFOR	2022-11-28	26131	24830	1553	U-238	172	11.08
2	2022-12-12	EXFOR	2022-11-28	26131	24830	1553	C-12	118	7.6
3	2022-12-12	EXFOR	2022-11-28	26131	24830	1553	Th-232	106	6.83
4	2022-12-12	EXFOR	2022-11-28	26131	24830	1553	H-2	90	5.8

SELECT statement. EXFOR statistics: Quantities

```
select date() as Today,'EXFOR' as DB
,(select max(date(UpdateDate)) from X4UPDATE) as Version
,(select count(EntryID) as nEntries from ENTRY) as nEntryAll
,(select count(distinct EntryID) from SUBENT where DnRow>0) as nEntryDat
,REACSTR.Quant,count(distinct SUBENT.EntryID) as nEntry
,round(count(distinct SUBENT.EntryID)*100./(select count(EntryID) from ENTRY),1) as Percent
,QUANTITY.ShortHelp
from REACSTR
inner join SUBENT on SUBENT.SubentID=REACSTR.SubentID
inner join QUANTITY on QUANTITY.Code=REACSTR.Quant -- try left join
group by REACSTR.Quant
order by nEntry desc
```

```
1  select date() as Today,'EXFOR' as DB
2  ,(select max(date(UpdateDate)) from X4UPDATE) as Version
3  ,(select count(EntryID) as nEntries from ENTRY) as nEntryAll
4  ,(select count(distinct EntryID) from SUBENT where DnRow>0) as nEntryDat
5  ,REACSTR.Quant,count(distinct SUBENT.EntryID) as nEntry
6  ,round(count(distinct SUBENT.EntryID)*100./(select count(EntryID) from ENTRY),1) as Percent
7  ,QUANTITY.ShortHelp
8  from REACSTR
9  inner join SUBENT on SUBENT.SubentID=REACSTR.SubentID
10 inner join QUANTITY on QUANTITY.Code=REACSTR.Quant
11 group by REACSTR.Quant
12 order by nEntry desc
```

	Today	DB	Version	nEntryAll	nEntryDat	Quant	nEntry	Percent	ShortHelp
1	2022-12-11	EXFOR	2022-11-28	26131	24830	CS	12866	49.2	Cross section data
2	2022-12-11	EXFOR	2022-11-28	26131	24830	DAP	4828	18.5	Partial differential data with respect to angle
3	2022-12-11	EXFOR	2022-11-28	26131	24830	DA	4720	18.1	Differential data with respect to angle
4	2022-12-11	EXFOR	2022-11-28	26131	24830	RP	2139	8.2	Resonance parameters

```
Execution finished without errors.
Result: 27 rows returned in 1361ms
At line 1:
select date() as Today,'EXFOR' as DB
```

SELECT statement. Search for UPPER case TITLE.

```
select distinct SUBENT.Entry,SUBENT.DateCompil
,ENTRY.Reference1,KEYWORD.FreeText as Title
 from SUBENT
 inner join ENTRY on ENTRY.EntryID=SUBENT.EntryID
 inner join KEYWORD on KEYWORD.EntryID=SUBENT.EntryID
 where SUBENT.SubAcc like '%001'
 and ((SUBENT.Entry like '2%') or (SUBENT.Entry like 'O%'))
 and KEYWORD.KeyWord='TITLE'
 and upper(KEYWORD.FreeText)=KEYWORD.FreeText
 order by SUBENT.Entry
```

SQLite.
Search Entries with UPPER case Title from Area 2 compiled between 1990 and 1999


```
select distinct SUBENT.Entry,SUBENT.DateCompil
,ENTRY.Reference1,KEYWORD.FreeText as Title
 from SUBENT
 inner join ENTRY on ENTRY.EntryID=SUBENT.EntryID
 inner join KEYWORD on KEYWORD.EntryID=SUBENT.EntryID
 where SUBENT.SubAcc like '%001'
 and ((SUBENT.Entry like '2%') or (SUBENT.Entry like 'O%'))
 and KEYWORD.KeyWord='TITLE'
 and upper(KEYWORD.FreeText)=KEYWORD.FreeText
 order by SUBENT.Entry
```


	Entry	DateCompil	Reference1	Title
1	20080	1983-09-01	J,OAW,174,575,6601	-REACTION CROSS-SECTION OF SR-90 WITH ...
2	20081	1983-09-01	J,OAW,175,145,6603	-THE EXISTENCE OF A K-39(N,2ALPHA) ...
3	20082	1983-09-01	J,OAW,176,227,68	-THE (N,N GAMMA) REACTIONS IN BR-79, Y-89, ...
4	20084	1983-09-01	J,OAW,176,311,6806	-THE ROLE OF THE ISOSPIN IN (N,2N)-...
5	20085	1983-09-01	J,OAW,176,45,6703	-GENERATION OF SOME SHORT LIVING ...
6	20086	1983-09-01	J,OSA,95,199,5811	-MEASUREMENT OF THE (N,ALPHA) AND (N,N ...

```
Execution finished without errors.
Result: 1179 rows returned in 782ms
At line 1:
select distinct
```

```
-THE (N,N GAMMA) REACTIONS IN BR-79, Y-89, YB-176 AND
AU-197 IN CONNECTION WITH THE PROBLEM OF THE
(N,2N) REACTION MECHANISM.- (IN GERMAN).
```

Type of data currently in cell: Text / Numeric
146 characters

Apply

Datasets having KEYWORD ‘Detector’ and ‘Method’

```
SELECT DISTINCT REACODE.ReacodeID,KWD.KeyWord as kw1,KWD.Code kw1c
, KWM.KeyWord as kw2,KWM.Code as kw2c, ENTRY.YearRef1, ENTRY.Author1, ENTRY.nAuthors
, ENTRY.Reference1, REACODE.nReacstr, REACODE.fullCode, REACODE.Pointer
, REACODE.nDataLines, REACODE.zaTarget1, REACODE.zaIncident1, REACODE.MF, REACODE.MT, REACODE.eMin, REACODE.eMax
 FROM  ENTRY INNER JOIN SUBENT ON ENTRY.EntryID=SUBENT.EntryID
 INNER JOIN REACSTR ON SUBENT.SubentID=REACSTR.SubentID
 INNER JOIN REACODE ON REACSTR.ReacodeID=REACODE.ReacodeID
 INNER JOIN KEYWORD AS KWD ON ENTRY.EntryID=KWD.EntryID
 INNER JOIN KEYWORD AS KWM ON ENTRY.EntryID=KWM.EntryID
 WHERE (REACSTR.Target='Al-27') AND (REACSTR.Quant='CS')
 AND (((KWD.SubAccNum=1)OR(KWD.SubentID=SUBENT.SubentID)))
 AND ((KWD.KeyWord='DETECTOR')AND(KWD.Code='SCIN'))
 AND (((KWM.SubAccNum=1)OR(KWM.SubentID=SUBENT.SubentID)))
 AND ((KWM.KeyWord='METHOD')AND(KWM.Code='ACTIV'))
 ORDER BY REACODE.zaTarget1,REACODE.nReacstr,REACODE.fullCode ,ENTRY.YearRef1 DESC,SUBENT.SubAcc
```

Search Datasets having Keywords:
DETECTOR with 'SCIN'
and METHOD with 'ACTIV'
INNER JOIN table KEYWORD twice
with different aliases

```
1   SELECT DISTINCT REACODE.ReacodeID,KW_Detector.KeyWord as kw1,KW_Detector.Code kw1c
2   , KW_Method.KeyWord as kw2,KW_Method.Code as kw2c, ENTRY.YearRef1, ENTRY.Author1, ENTRY.nAuthors
3   , ENTRY.Reference1, REACODE.nReacstr, REACODE.fullCode, REACODE.Pointer
4   , REACODE.nDataLines, REACODE.zaTarget1, REACODE.zaIncident1, REACODE.MF, REACODE.MT, REACODE.eMin, REACODE.eMax
5    FROM  ENTRY INNER JOIN SUBENT ON ENTRY.EntryID=SUBENT.EntryID
6    INNER JOIN REACSTR ON SUBENT.SubentID=REACSTR.SubentID
7    INNER JOIN REACODE ON REACSTR.ReacodeID=REACODE.ReacodeID
8    INNER JOIN KEYWORD AS KW_Detector ON ENTRY.EntryID=KW_Detector.EntryID
9    INNER JOIN KEYWORD AS KW_Method ON ENTRY.EntryID=KW_Method.EntryID
10   WHERE (REACSTR.Target='Al-27') AND (REACSTR.Quant='CS')
11   AND (((KW_Detector.SubAccNum=1)OR(KW_Detector.SubentID=SUBENT.SubentID)))
12   AND ((KW_Detector.KeyWord='DETECTOR')AND(KW_Detector.Code='SCIN'))
13   AND (((KW_Method.SubAccNum=1)OR(KW_Method.SubentID=SUBENT.SubentID)))
14   AND ((KW_Method.KeyWord='METHOD')AND(KW_Method.Code='ACTIV'))
15   ORDER BY REACODE.zaTarget1,REACODE.nReacstr,REACODE.fullCode ,ENTRY.YearRef1 DESC,SUBENT.SubAcc
```

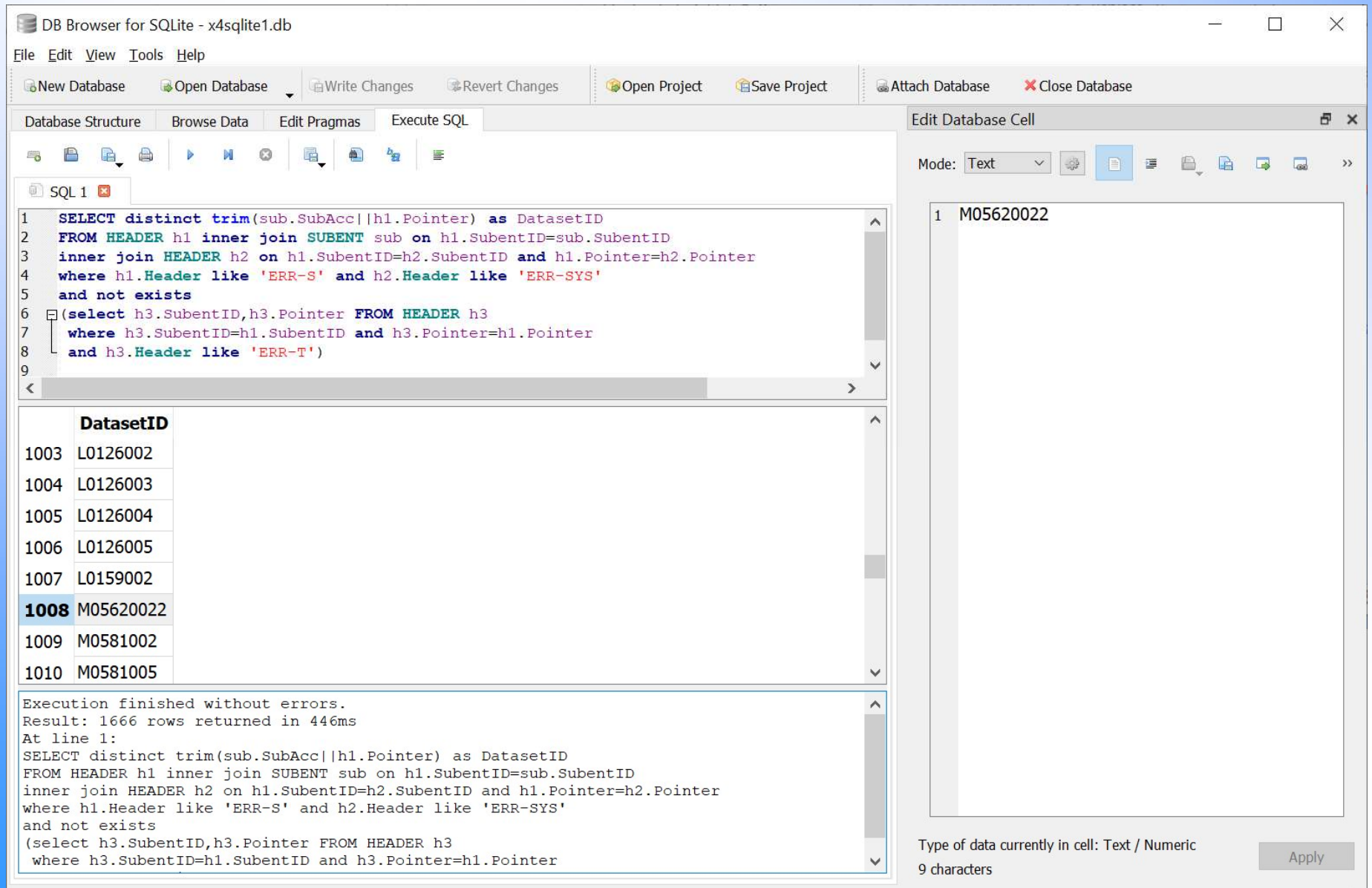
	ReacodeID	kw1	kw1c	kw2	kw2c	YearRef1	Author1	nAuthors	Reference1	nReacstr	fullCode	Pc
1	218460351	DETECTOR	SCIN	METHOD	ACTIV	1970	Schantl	1	T,SCHANTL,1970	2	(8-O-16(N,P)7-N-16,,SIG)/(13-...	1
2	M0267002	DETECTOR	SCIN	METHOD	ACTIV	1965	Thompson	4	J,NP,64,486,1965	1	13-AL-27(G,N)13-AL-26-...	
3	41614005	DETECTOR	SCIN	METHOD	ACTIV	2016	Filatenkov	1	R,INDC(CCP)-0460,2016	1	13-AL-27(N,2N)13-AL-26-G,,SIG	
4	41240073	DETECTOR	SCIN	METHOD	ACTIV	1999	Filatenkov	17	R,RI-252,1999	1	13-AL-27(N,2N)13-AL-26-G,,SIG	

```
Execution finished without errors.
Result: 74 rows returned in 580ms
At line 1:
SELECT DISTINCT REACODE.ReacodeID,KW_Detector.KeyWord as kw1,KW_Detector.Code kw1c
,KW_Method.KeyWord as kw2,KW_Method.Code as kw2c
```

Sub-query. Search Headers ERR-S, ERR-SYS, ERR-T

```
SELECT distinct trim(sub.SubAcc||h1.Pointer) as DatasetID
FROM HEADER h1 inner join SUBENT sub on h1.SubentID=sub.SubentID
inner join HEADER h2 on h1.SubentID=h2.SubentID and h1.Pointer=h2.Pointer
where h1.Header like 'ERR-S' and h2.Header like 'ERR-SYS'
and not exists (select h3.SubentID,h3.Pointer FROM HEADER h3
 where h3.SubentID=h1.SubentID and h3.Pointer=h1.Pointer
 and h3.Header like 'ERR-T')
```

SQLite.
Search Datasets with ERR-S and ERR-T but without ERR-T



SELECT statement with JSON_EXTRACT()

```
select json_extract(xdat,'$.EN') as En
,json_extract(xdat,'$.ANG') as Ang
,json_extract(xdat,'$.DATA-CM') as Dat
,xdat
from x4pro_x4data
where DatasetID='A1495003'
and Ang=150 -- MariaDB: and json_extract(xdat,'$.ANG')=150
order by En -- MariaDB: order by json_extract(xdat,'$.EN')
```

```
create table x4pro_x4data (
    DatasetID   varchar(9) not null,
    idat        integer null,
    xdat        JSON null,
    primary key (DatasetID,idat)
)
```

Database Structure | Browse Data | Edit Pragmas | Execute SQL

SQL 1

```
1  select json_extract(xdat,'$.EN') as En
2  ,json_extract(xdat,'$.ANG') as Ang
3  ,json_extract(xdat,'$.DATA-CM') as Dat
4  ,xdat
5  from x4pro_x4data
6  where DatasetID='A1495003' and Ang=150
7  order by En
```

	En	Ang	Dat	xdat
1	0.8989	150.0	0.7892	{"DATA-CM":0.7892,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":...
2	0.9216	150.0	0.8881	{"DATA-CM":0.8881,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":...
3	0.9518	150.0	1.036	{"DATA-CM":1.036,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":...
4	0.9554	150.0	1.135	{"DATA-CM":1.135,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":...
5	0.9971	150.0	1.271	{"DATA-CM":1.271,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":...
6	1.008	150.0	1.357	{"DATA-CM":1.357,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":...
7	1.05	150.0	1.531	{"DATA-CM":1.531,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":...
8	1.088	150.0	1.716	{"DATA-CM":1.716,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":...
9	1.125	150.0	1.95	{"DATA-CM":1.95,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":...

```
Execution finished without errors.
Result: 99 rows returned in 4ms
At line 1:
select json_extract(xdat,'$.EN') as En
,json_extract(xdat,'$.ANG') as Ang
,json_extract(xdat,'$.DATA-CM') as Dat
,xdat
from x4pro_x4data
where DatasetID='A1495003' and Ang=150
order by En
```

Edit Database Cell

Mode: Text

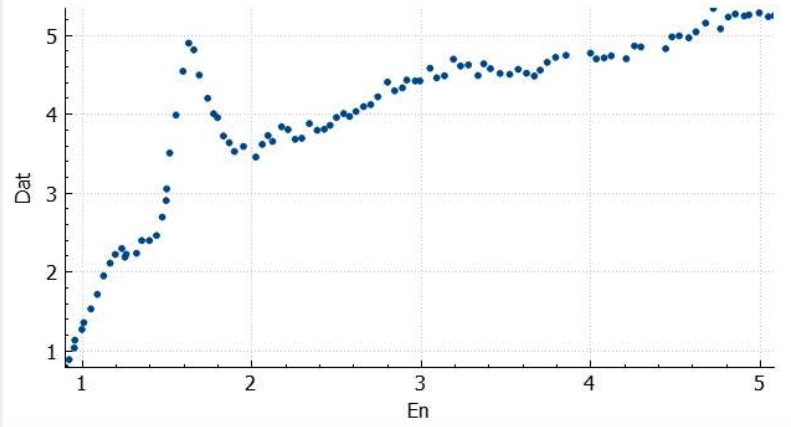
```
1  {"DATA-CM":1.036,"DATA-ERR":20.0,"ERR-DIG":0.012,"EN":0.9
   518,"EN-ERR-DIG":0.004,"E-LVL":2.9,"ANG":150.0}
```

Type of data currently in cell: Valid JSON
104 characters

Apply

Plot

Columns	X	Y1	Y2	Axis Type
Row...	☐	☐	☐	Numeric
En	☑	☐	☐	Numeric
Ang	☐	☐	☐	Numeric
Dat	☐	☑	☐	Numeric
xdat	☐	☐	☐	Label



Line type: None Point shape: Disc

UTF-8

Python

High-level, general-purpose programming language

Wikipedia: “Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming”, since ~1990

Python examples

Hello, World!

```
print('Hello, world') #line comment
```

How to run it

```
$ python p1.py
Hello, world
```

How to run it

```
$ python3 p1.py
Hello, world
```

Connect to SQLite database, execute SQL statement. Note: sqlite3 comes with python3

```
import sqlite3
try:
    conn=sqlite3.connect("x4sqlite1.db",uri=True)
    conn.row_factory=sqlite3.Row  #in order to access data by column names
except Error:
    print(Error)
cursor=conn.cursor()
sql="select Entry,YearRef1,Author1 from ENTRY where Entry like 'F%'"
try:
    cursor.execute(sql)
    rows=cursor.fetchall()
except Error:
    print(Error)
cursor.close()
conn.close()
ii=0
for row in rows:
    Entry=row[0]
    Entry=row['Entry']   #access data by column name
    YearRef1=row[1];  Author1=row[2] #several instructions in one line separated by <;>
    ii+=1
    print('\t'+str(ii)+') '+str(Entry)+' '+str(YearRef1)+' '+Author1)
```

```
$ python db0.py
          1) F0001 1976 Gould
          2) F0002 1968 Ludecke
          3) F0003 1953 Sawyer
          4) F0004 1975 Liskien
          5) F0005 1979 Shinozuka
          . . . . .
```

Fortran

General-purpose, imperative programming language

Wiki: “especially suited to numeric computation and scientific computing”, since ~1954

Fortran examples

Hello, World! File: f0.f

```
      write (*,*) 'Hello, world'
      end
```

How to run it

```
$ gfortran -o f0 f0.f
$ f0
Hello, world
```

File f1.f : fonnect to SQLite database, execute SQL statement, call subroutine reading results

```
      program f1
      character*512  dbname
      character*512  outfile/'sql1tmp.dat'/
      character*4096 sql
      write (*,*) 'Program: f1.f (ver.2022-12-12)'
      write (*,*) 'by V.Zerkin, IAEA-NDS, 2021-2022'
      dbname='../../x4sqlite1.db'
      ierr=ix4lite_open(trim(dbname)//char(0))
      !--- char(0) added for compatibility with C
      if (ierr.ne.0) then
          write (*,*) 'Can''t open database'
          stop
      endif
      write (*,*) 'Open database: ',trim(dbname),' ierr=',ierr
      write (*,*)
      sql='select Entry,YearRef1,Author1'
     1  //' from ENTRY where Entry like ''F%'''
      write (*,*) 'SQL command:'
      write (*,*) trim(sql)
      sql=trim(sql)//char(0)
      outfile=trim(outfile)//char(0)
      ipnt=ix4lite_exec(sql,outfile)
      if (ierr.lt.0) then
          write (*,*) 'SQL error ierr=',ipnt
      else
          write (*,*) 'SQL executed OK:',ipnt,' rows'
      endif
      call read_data(outfile,ipnt)
      call ix4lite_close();
      write(*,'(/a)') 'Program completed successfully'
      stop
      end
```

Traditional Fortran-SQL problem: dealing with data types, pointers, C-style data and programming.

My approach for X4Pro: use simple C subroutines writing result of SQL to temporary text file and read result in Fortran code using formatted input. No pointers, almost no C.

Temporary file with result of SQL

```
#### 1
$Entry       F0001
$YearRef1    1976
$Author1     Gould
//
#### 2
$Entry       F0002
$YearRef1    1968
$Author1     Ludecke
//
#### 3
$Entry       F0003
$YearRef1    1953
$Author1     Sawyer
//
. . . . . . . . .
#### 1427
$Entry       F1464
$YearRef1    1990
$Author1     Ignatenko
//
//EOF
```

Fortran examples

Continue file f1.f : Subroutine reading results from temporary file

```
      subroutine read_data(infile,ipnt)
      character(len=*) infile
      character*132 nam,line
      character*5   Entry
      character*4   sYear
      character*12  Author1
      write (*,*) 'Read data points:',ipnt
      nin=100
      open(unit=nin,file=infile,status='old',err=9999)
      iipnt=0
      do
          read(nin,'(a12,a)',end=8888) nam,line
          if (nam.eq.'//EOF')       exit
          if (nam.eq.'$Entry')      Entry=line
          if (nam.eq.'$YearRef1')   read(line,'(i11)') iYear
          if (nam.eq.'$YearRef1')   sYear=line
          if (nam.eq.'$Author1')    Author1=line
          if (nam(1:4).eq.'####') then !--- Begin of new Row
             iipnt=iipnt+1
             Entry=''
             iYear=0
             sYear='----'
             Author1=''
             cycle
          endif
          if (nam.eq.'//') then !--- End of Row
             write(*,*) iipnt,Entry,' ',sYear,',',Author1
          endif
      enddo
8888  close(nin)
9999  return
      end
```

How to compile it. Note: sqlite3.c is provided by SQLite developers team.

```
          #Windows:
$ gfortran f1.f sql1sub.c sqlite3.c -o f1
          #Linux, MacOS:
$ gfortran f1.f sql1sub.c sqlite3.c -o f1.exe -pthread -ldl
```

Temporary file with result of SQL

```
#### 1
$Entry       F0001
$YearRef1    1976
$Author1     Gould
//
#### 2
$Entry       F0002
$YearRef1    1968
$Author1     Ludecke
//
#### 3
$Entry       F0003
$YearRef1    1953
$Author1     Sawyer
//
. . . . . .
#### 1427
$Entry       F1464
$YearRef1    1990
$Author1     Ignatenko
//
//EOF
```

How to run it

```
$ f1
$ ./f1
 Program: f1.f (ver.2022-12-12)
 by V.Zerkin, IAEA-NDS, 2021-2022
 Open database: ../../x4sqlite1.db ierr=0

 SQL command:
 select Entry,YearRef1,Author1
 from ENTRY where Entry like 'F%'
 SQL executed OK:         1427  rows
 Read data points:         1427
           1 F0001 1976,Gould
           2 F0002 1968,Ludecke
           3 F0003 1953,Sawyer
```

Fortran examples. C subroutines

File sql1sub.c : connect to SQLite database, execute SQL statement, store results to temporary file

```
//gcc -c sql1sub.c
// Copyright (C) 2021-2022 International Atomic Energy Agency (IAEA)
// Author: Viktor Zerkin, V.Zerkin@iaea.org

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "sqlite3.h"
static int irec=0;
static sqlite3 *db=NULL;
static const char* data="Callback function called";
static FILE *outFile=NULL;
static char sErrMsg[512];

static int callback(void *data, int argc, char **argv, char **azColName) {
    int i;
    irec++;
    if (outFile!=NULL) fprintf(outFile,"#### %d\n",irec);
    else printf("#%d          %s\n",irec,(const char*)data);
    for (i=0; i<argc; i++) {
         if (outFile!=NULL) {
             if (argv[i]!=NULL) fprintf(outFile,"$%-10s %s\n",azColName[i], argv[i]);
         }
         else
         printf("%-10s : %s\n", azColName[i], argv[i] ? argv[i] : "NULL");
    }
    if (outFile!=NULL) fprintf(outFile,"//\n");
    else printf("\n");
    return 0;
}

int ix4lite_open_(const void *filename) {
    int ierr;
    if (db!=NULL) sqlite3_close(db);
    irec=0;
    ierr=sqlite3_open_v2(filename,&db,SQLITE_OPEN_READONLY,NULL);
    if (ierr!=0) {
         sprintf(sErrMsg,"Can't open database: %s\n",sqlite3_errmsg(db));
    } else {
//       fprintf(stderr,"Opened database successfully\n");
    }
    return ierr;
}

int ix4lite_close_() {
    int ierr;
    if (db!=NULL) sqlite3_close(db);
    return 0;
}
```

```
int ix4lite_exec_(char *sql,const void *filename)
{
    int ierr;
    char *zErrMsg=0;
    if (db==NULL) return -3;

    if (strcmp(filename,"tt")!=0) {
         outFile=fopen(filename,"w");
         if (outFile==NULL) {
            fprintf(stderr,
"Can not open output file [%s]\n",(char *)filename);
             return -2;
         }
    }
    //Execute SQL statement
    ierr=sqlite3_exec(db,sql,callback,(void*)data,
         &zErrMsg);
    if (ierr!=SQLITE_OK) {
         sprintf(sErrMsg,"SQL error: %s\n",zErrMsg);
         printf("SQL error: %s\n",zErrMsg);
         sqlite3_free(zErrMsg);
         return -1;
    }
    if (outFile!=NULL) {
         fprintf(outFile,"//EOF\n");
         fclose(outFile);
    }

    return irec;
}

void getErrMsg_(char *errMsg)
{
    sprintf(errMsg,"%s",sErrMsg);
}
```

Concluding remarks

1. Basic language needed to deal with X4Pro: **SQL**

   Because X4Pro is distributed from Nuclear Data Center end-user mostly operates with SELECT statement.

2. A language for programming of X4Pro can be any (having SQL support)

   Examples of programming are given in Python and Fortran

3. X4Pro stores part of data in JSON form

   This data can be accessed also as basic type data using functions json_extract(), json_value(), etc. presented in SQLite and MariaDB

Thank you.