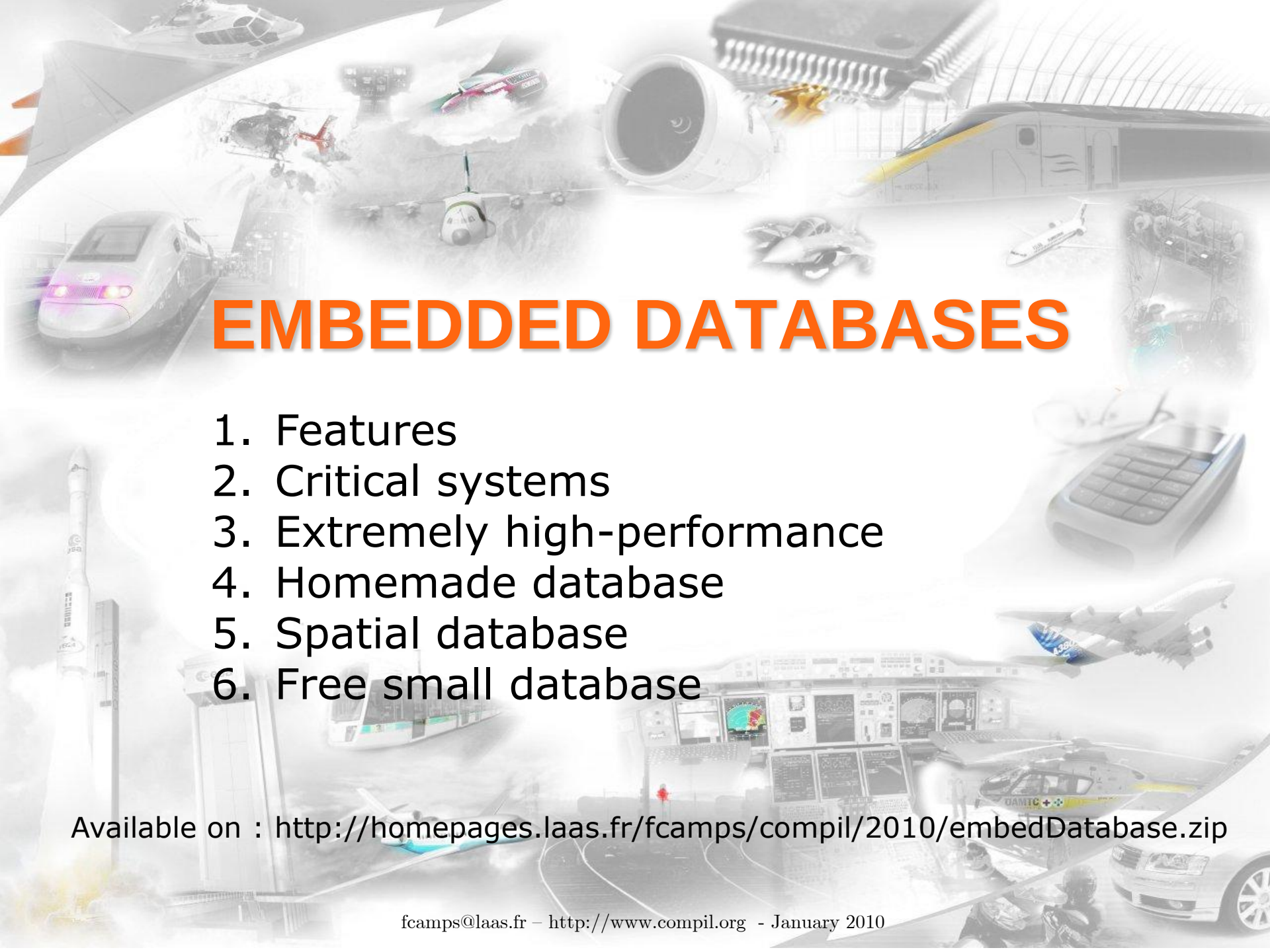


EMBEDDED DATABASES

1. Features
2. Critical systems
3. Extremely high-performance
4. Homemade database
5. Spatial database
6. Free small database

Available on : http://homepages.laas.fr/fcamps/compil/2010/embedDatabase.zip

EMBEDDED DATABASES

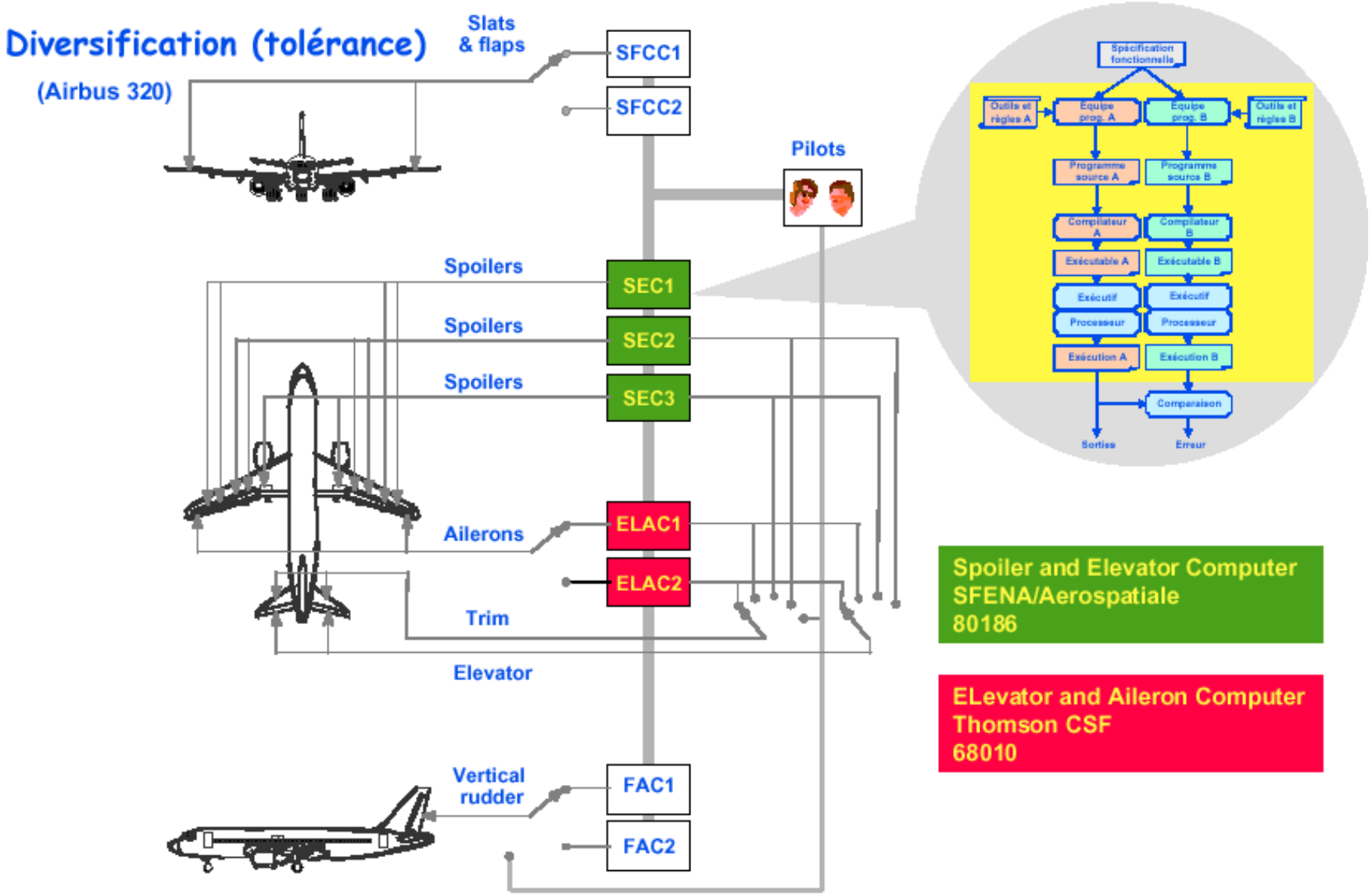
- Embedded systems ?
- Common features for many database
  - What is a system without data?
  - A system needs data input / output to provide a service
  - What the embedded system do with its data ? :
    - Data are used to run the systems itself
      - GPS receiver
      - Airplane : autopilot, radar information
      - ABS, cruise control in automotive
      - Heart pace maker (health care)
    - Use of sensors, M2M interfaces
  - Data are needed to take decision and to make system reactive

EMBEDDED DATABASES / Features

- The embedded system :
  - Critical (Certified code) and non critical embedded systems
  - Systems with and whitout OS
  - Real time operating system with real time software

- The embedded databases
  - Can’t store a large or small amount of data
  - Reduced maintenance
  - Robustness and resilience
  - Real-time behaviour
  - Management synchronous / asynchronous
  - Certified code
  - Performance
  - Small footprint

EMBEDDED DATABASES / Critical systems



Spoiler and Elevator Computer
SFENA/Aerospatiale
80186

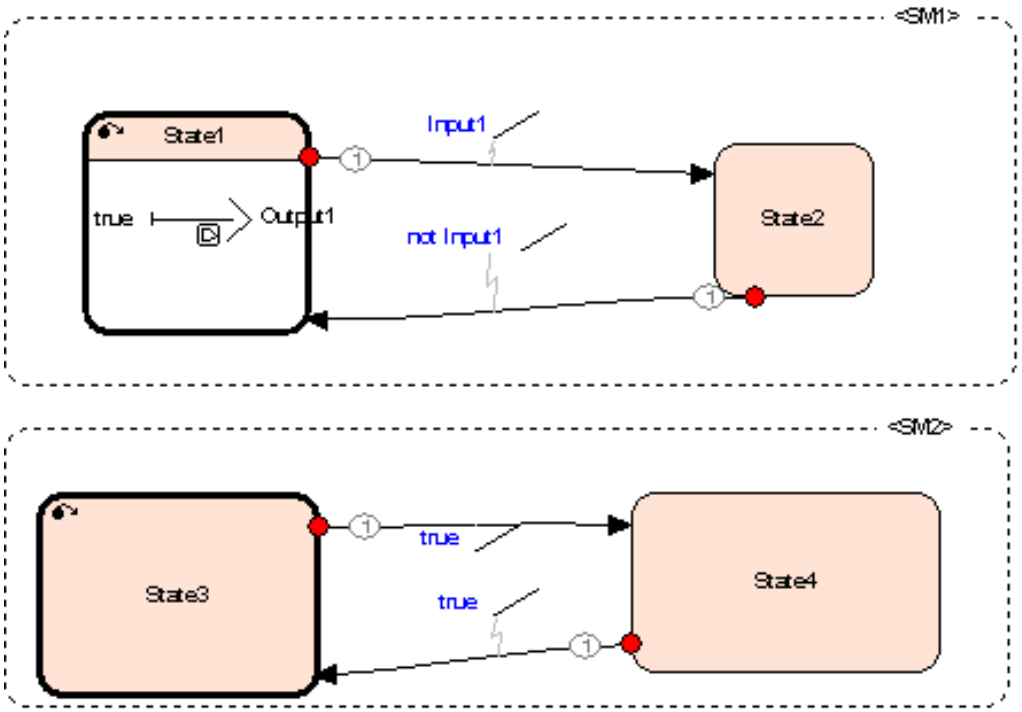
ELevator and Aileron Computer
Thomson CSF
68010

EMBEDDED DATABASES / Critical systems

- Formal proof
  - Formal proofs must be supplied for extremely critical aspects (eg an AND gate)
- Test coverage
  - All generated code must be executed
- Certified code (for example DO178B, EN50128 for railways)
  - The code should follow a certification process (coding rule very strictly, replay code, unit testing and integration, simulation)
  - Do not "mix" of the code certified and uncertified
- Specific language
  - Some languages are more suited to the development of complex systems using databases as memory or file
  - ADA, Lustre, Esterelle, SCADE (Lustre + Esterelle), C
  - **Model based design / certified software factory**

EMBEDDED DATABASES / Critical systems

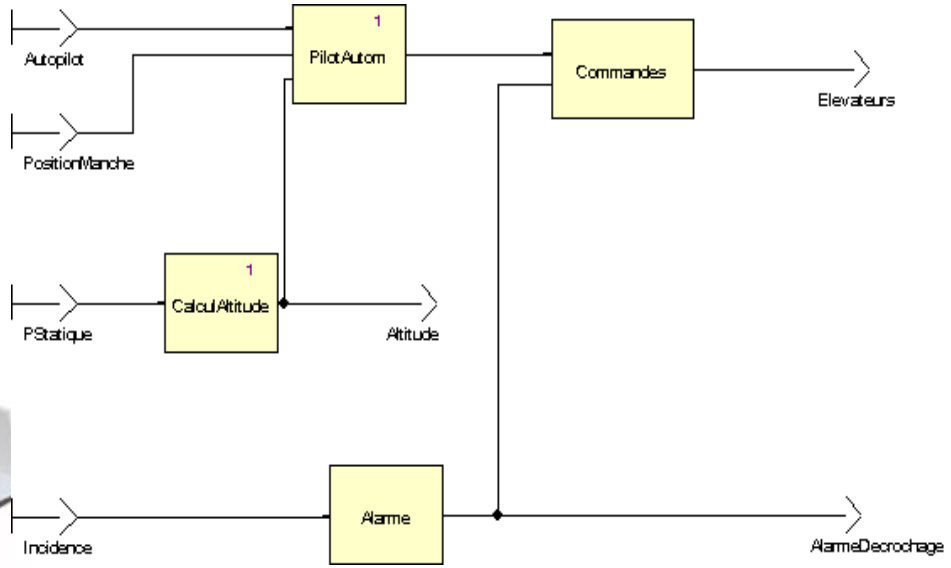
- Code development : example with language SCADE
- The database & OS are certified so we have to produce certified code
- Load data in memory from disk - Data are updated by process (parallel simple state machine with mutual access exclusion : demo)



Source/Target	Conditions/Actions
Source: SM1:State1 **Target:** SM1:State2	**Condition:** Input1
Source: SM1:State2 **Target:** SM1:State1	**Condition:** not Input1
Source: SM2:State3 **Target:** SM2:State4	**Condition:** true
Source: SM2:State4 **Target:** SM2:State3	**Condition:** true

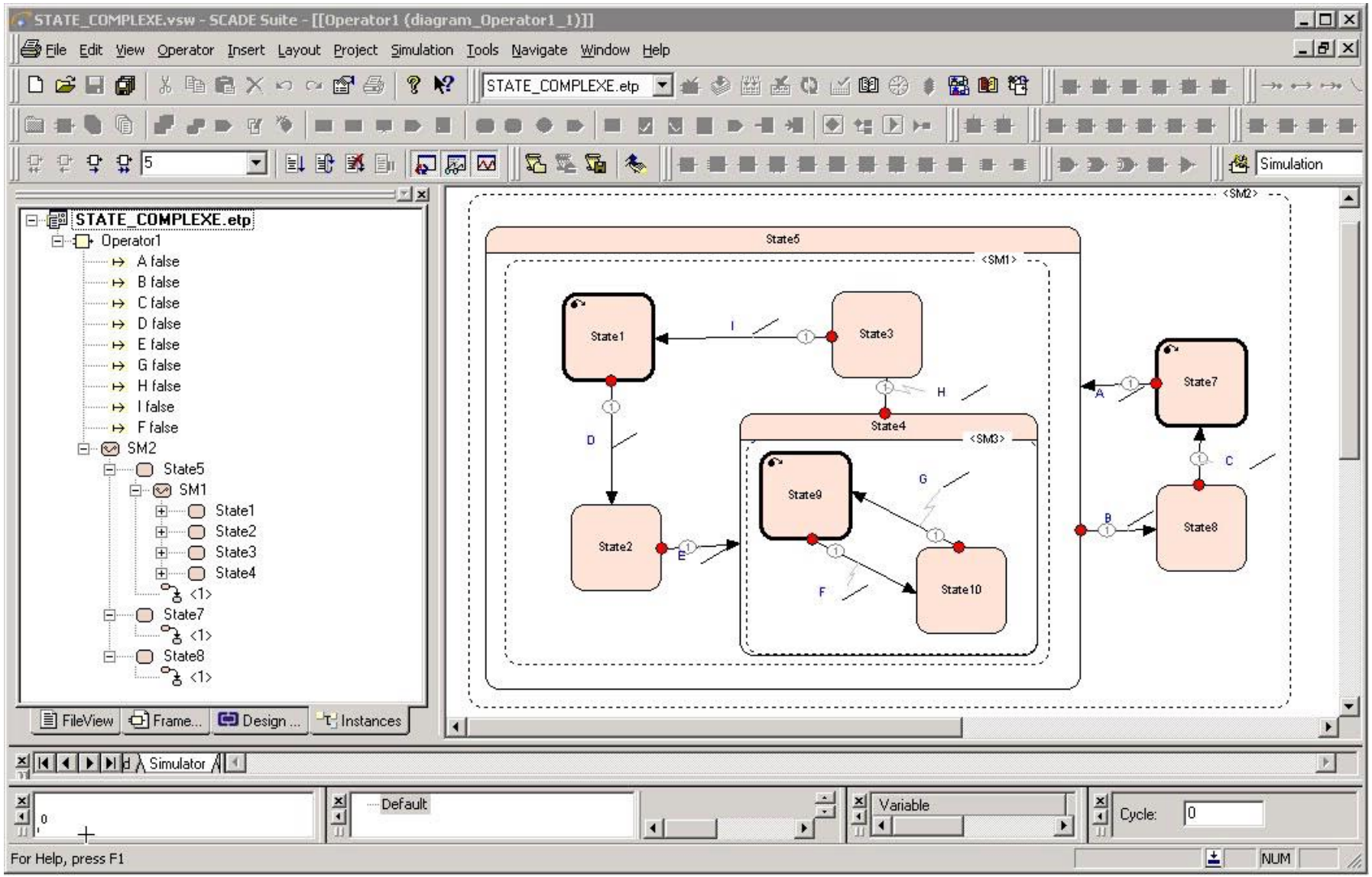
EMBEDDED DATABASES / Critical systems

- A380 autopilot with SCADE (15K diagram operators)
- In-memory database is use with certified operators
- Functions : Calculation of flight parameters, alarm, flight controls, autopilot



EMBEDDED DATABASES / Critical systems

- Example SCADE : inside operator parallel complex state machine (demo)



- Example SCADE : code generation for a logical AND operator

```
#include <stdio.h>
#include "Operator1.h "

//  Operateur1  AND :
//                 _______
//  Input1 ->|             |
//           |             |---> Output1
//           |             |
//  Input2 ->|_____________|
//
//

int main()
{

printf("Test scade operator 1\n");

// structure de l'operateur
inC_Operator1  in; // les entrees
outC_Operator1 out; // les sorties

// reset de l'operateur1
Operator1_reset(&out);
```

```
...
////////////////////////////////
// test des entrees tout a "false"
in.Input1=kcg_false; // Input1 = false
in.Input2=kcg_false; //Input2 = false
Operator1(&in, &out); // application à l'operateur
// sortie de l'operateur
printf("output operator : %d\n", out.Output1);

////////////////////////////////
// test des entrees "true" + "false"
in.Input1=kcg_false;
in.Input2=kcg_true;
Operator1(&in, &out);
// sortie de l'operateur
printf("output operator : %d\n", out.Output1);

return 0;
}

// resultat du programme :
C:\tmp\test1\KCG>a.exe
Test scade operator 1
output operator : 1
output operator : 0
```

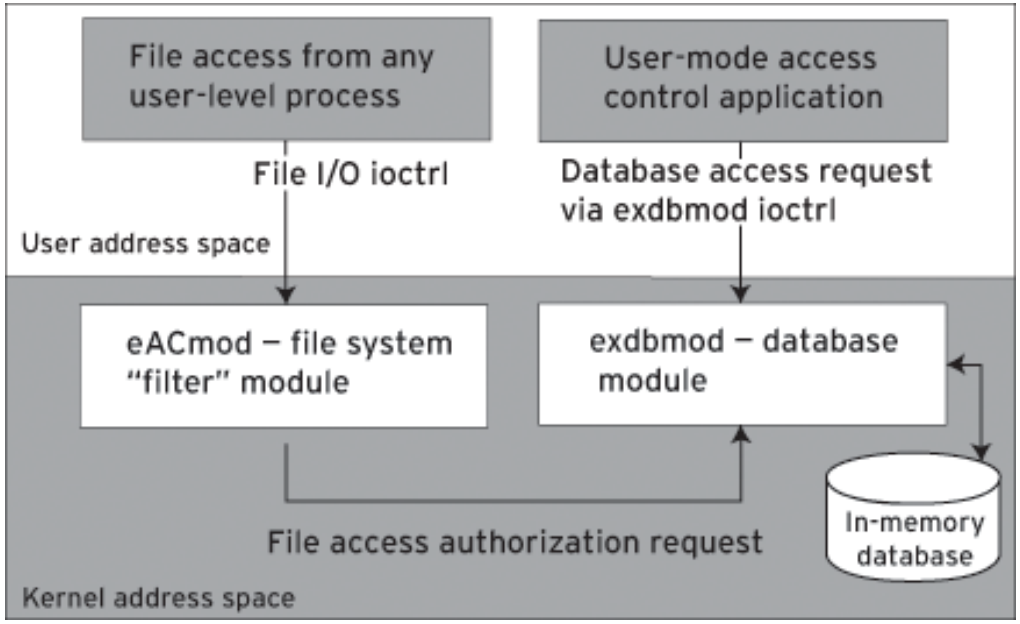
Database technology for extremely high-performance applications

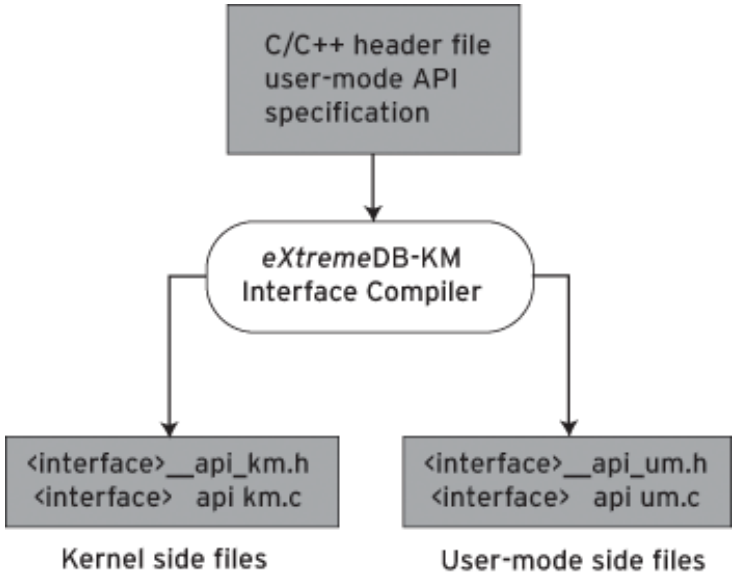
- What we ask for ?
  - Eliminating latency and increasing the prioritization of data management tasks.
  - The ability to store entire databases in main memory, eliminating the overhead and unpredictability of disk I/O and caching logic.
  - zero-latency responsiveness
  - ultra-small footprint
  - High level request langage (specialized API)
  - Critical and Determinism

Database technology for extremely high-performance applications

Possibilities ?

- **In-memory** (and disk) embedded database runtime adjusted for kernel usage.
- Database locking is implemented via kernel-mode spinlocks rather than synchronization primitives available in the user space (futex and extended futex)
- Linking the application to the database (reduce interprocess communication)
- The database kernel module (on real-time operating system ie VxWorks, XENOMAI)
- A user-mode application that utilizes a user-mode database API.



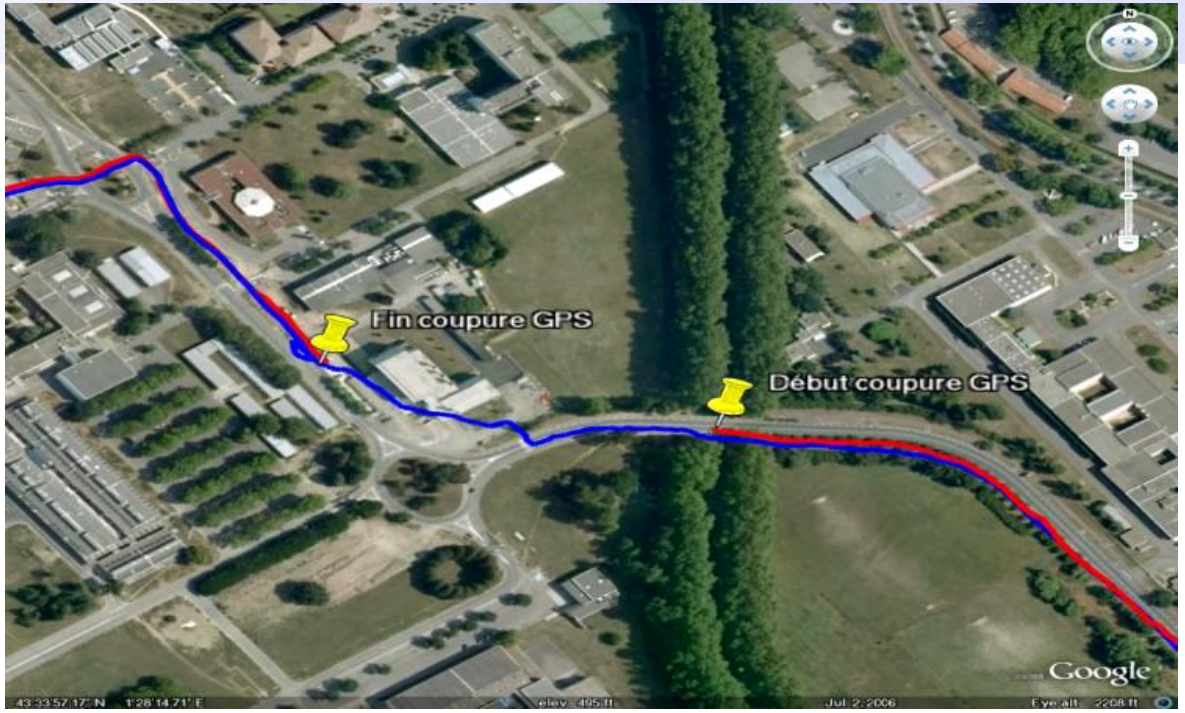


- **Considerations**
  - Not every application requiring high performance needs a kernel-mode database.
  - Drawbacks :
    - Portability
    - Fault protection : There is less room for error in the OS kernel compared to the user-mode environment
    - Faults caused by improper use of the database engine could render the kernel unusable and lead to system crashes
- **Conclusion**
  - Kernel-mode database in the OS kernel to accelerate overall system performance,
  - Keep the advantage of a full set of database features—including transaction processing, multithreaded data access, complicated querying using built-in indexing, data access API, and a high-level data definition

Homemade embedded database

visually impaired pedestrians help system

LAAS-CNRS

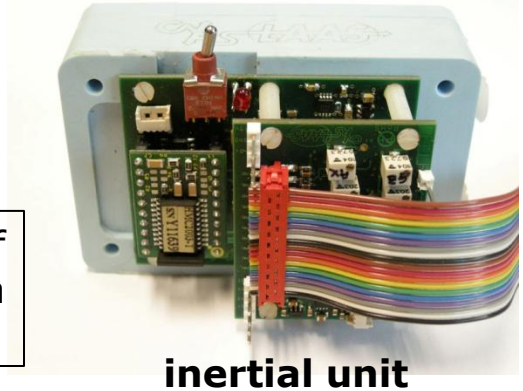


- In small embedded application (LAAS/CNRS BINAUR PROJECT)
  - In-memory data + file disk + simplified index
  - A process invokes a file read/write to record raw data, doesn't stay blocked until the OS has finished (whether successfully or unsuccessfully)
  - Side effect in RT
    - Any process can have pending I/O operation
    - It's a source of unpredictability

Homemade embedded database : file

- Asynchronous I/O - lib RT : SUSv3
  - Asynchronous functions revolve around asynchronous I/O control block
    - **struct aiocb** contains all information needed to describe I/O operation
    - **aio_read** and **aio_write** to schedule I/O operation
    - **aio_error** and **aio_return** retrieve the error and status after I/O completion
    - **aio_fsync** forces all I/O operations
    - **aio_suspend** to block the calling thread until I/O has been completed or timeout
    - **aio_cancel** cancels I/O not yet completed

To request the asynchronous notification of completion of operation either by signal or by asynchronous execution of a function



inertial unit

LAAS-CNRS

Homemade embedded database: file

```
#include <aio.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>
#include <sys/stat.h>

#define SIGNAL_IO      (SIGRTMIN + 3)

void schedule(int signum, siginfo_t * info, void * empty)
{
struct aiocb * cb;
ssize_t       nb_octets;
if (info->si_code == SI_ASYNCIO) {
                cb = info->si_value.sival_ptr;
                if (aio_error(cb) == EINPROGRESS)
                                return;
                nb_octets = aio_return(cb);
                fprintf(stdout, « Read 1 : %d octets read\n",
nb_octets);
}
}

void thread (sigval_t valeur)
{
struct aiocb * cb;
ssize_t       nb_octets;
cb = valeur.sival_ptr;
if (aio_error(cb) == EINPROGRESS)
                return;
nb_octets = aio_return(cb);
fprintf(stdout, « Read 2 : %d octets read\n", nb_octets);
}
```

Use RT signal

thread

IEEE 1003.1-2004

RT signal must be used carrefully

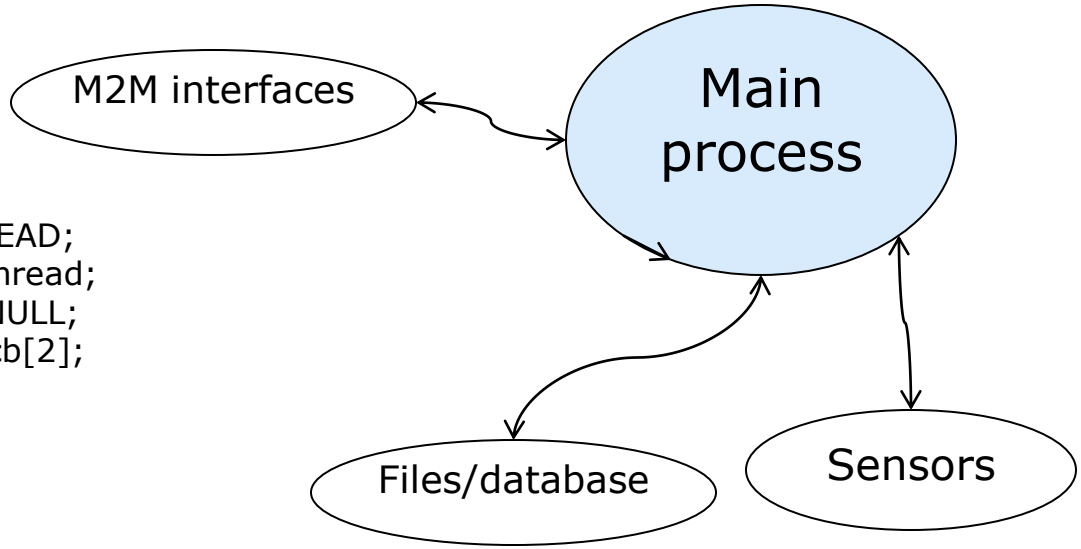
```
int
main (int argc, char * argv[])
{
int               fd;
struct aiocb      cb[3];
Char              buffer[256][3];
struct sigaction action;
int               nb_octets;

if (argc != 2) {
                fprintf (stderr, "Usage: %s file", argv[0]);
                exit(EXIT_FAILURE);
                }
if ((fd = open(argv[1], O_RDONLY)) < 0) {
                perror("open");
                exit(EXIT_FAILURE);
}
action.sa_sigaction = schedule;
action.sa_flags     = SA_SIGINFO;
sigemptyset(& action.sa_mask);
if (sigaction(SIGNAL_IO, & action, NULL) < 0) {
                perror("sigaction");
                exit(EXIT_FAILURE);
}

/* Read 0 :  no notify */
cb[0].aio_fildes    = fd;
cb[0].aio_offset    = 0;
cb[0].aio_buf       = buffer[0];
cb[0].aio_nbytes    = 256;
cb[0].aio_reqprio   = 0;
cb[0].aio_sigevent.sigev_notify = SIGEV_NONE;

/* Read 1 : notify by signal */
cb[1].aio_fildes    = fd;
cb[1].aio_offset    = 0;
cb[1].aio_buf       = buffer [1];
cb[1].aio_nbytes    = 256;
cb[1].aio_reqprio   = 0;
cb[1].aio_sigevent.sigev_notify = SIGEV_SIGNAL;
cb[1].aio_sigevent.sigev_signo  = SIGNAL_IO;
cb[1].aio_sigevent.sigev_value.sival_ptr = & cb[1];
```

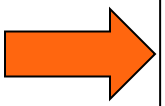
Async struct

Signal struct

Homemade embedded database : file

```
/* Read 2 : Notify by thread */
cb[2].aio_fildes    = fd;
cb[2].aio_offset    = 0;
cb[2].aio_buf       = buffer [2];
cb[2].aio_nbytes    = 256;
cb[2].aio_reqprio   = 0;
cb[2].aio_sigevent.sigev_notify = SIGEV_THREAD;
cb[2].aio_sigevent.sigev_notify_function    = thread;
cb[2].aio_sigevent.sigev_notify_attributes = NULL;
cb[2].aio_sigevent.sigev_value.sival_ptr = & cb[2];

/* start to read */
if ((aio_read (& cb[0]) < 0)
 || (aio_read (& cb[1]) < 0)
 || (aio_read (& cb[2]) < 0)) {
                perror("aio_read");
                exit(EXIT_FAILURE);
}
fprintf(stdout, "read launched\n");

while ((aio_error(& cb[0]) == EINPROGRESS)
|| (aio_error(& cb[1]) == EINPROGRESS)
||(aio_error(& cb[2]) == EINPROGRESS))
                sleep(1);
nb_octets = aio_return(& cb[0]);
fprintf(stdout, « Read O : %d octets read\n", nb_octets);
return EXIT_SUCCESS;
```





Process in progress while reading data

Homemade embedded database : memory

IEEE Std 1003.1-2004 standard specifies an interface to set up shared memory

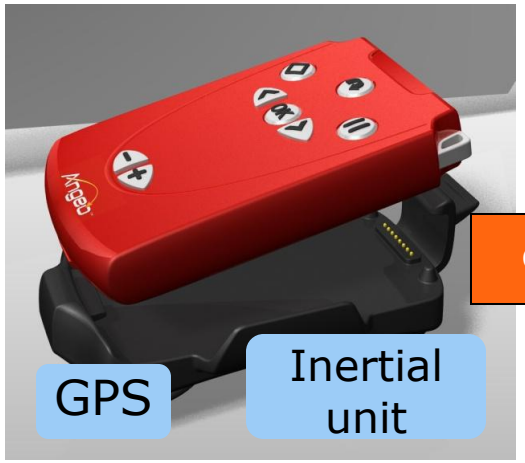
```
.....

 /*create semphore */
 if((semID=semget(key,nbSema,0666|IPC_CREAT)) < 0)
   {
     perror("semget");
   }

 ....
/* create shared memory */
 memoireID = shmget((key_t)key_memoire,sizeof(int), 0600 | IPC_CREAT);
 if (memoireID==-1)
   {
     perror("shmget");
     exit(EXIT_FAILURE);
   }

 /* attachement in user workspace */
 mem = (int*)shmat(memoireID,NULL,0); //acces read/write
 if ((int)mem==-1)
   {
     perror("shmat");
     exit(EXIT_FAILURE);
   }
```

semaphore

Shared memory as database

- See also : mmap in <sys/mman.h> + MAP_PRIVATE, MAP_SHARED, MAP_FIXED ...

Spatial embedded database



GSM

Visual ampaired pedestrian help system

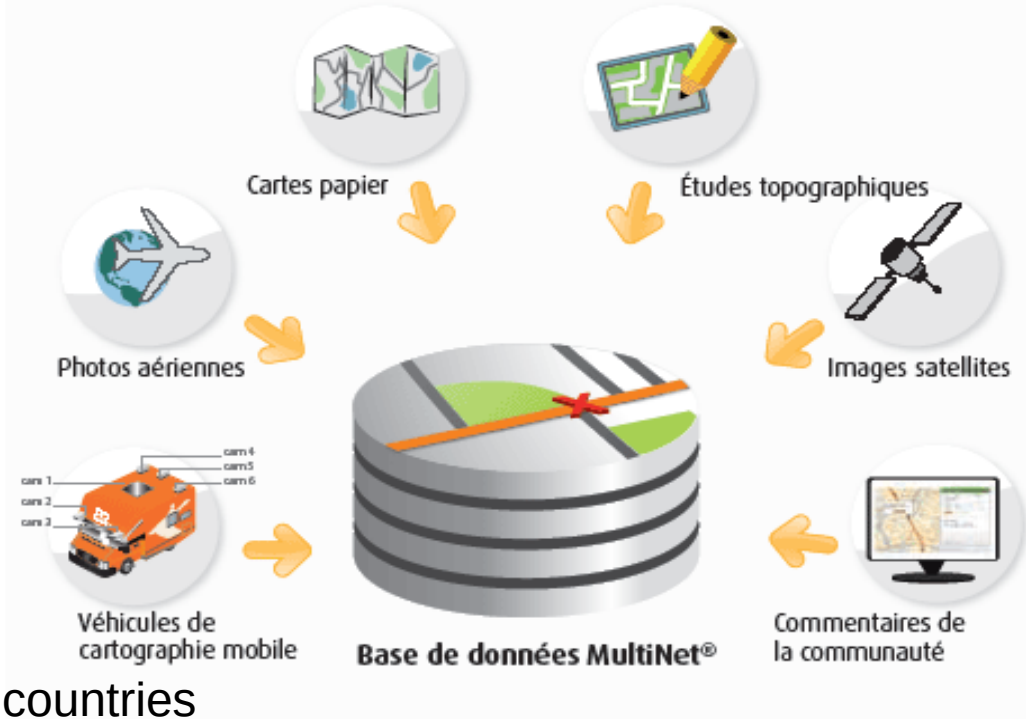
LAAS-CNRS

Tele Atlas MultiNet

An in-memory shapefile stores nontopological geometry and attribute information for the spatial features in a data set. (Lambert 93 projection).

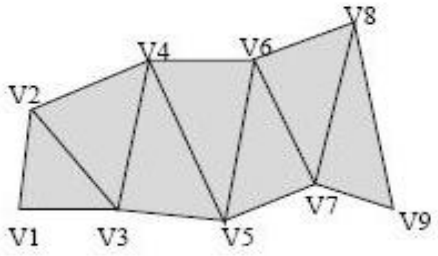
- .shp — shape format; the feature geometry itself
- .shx — shape index format; a positional index of the feature geometry to allow seeking forwards and backwards quickly
- .dbf — attribute format; columnar attributes for each shape, in dBase III format

Spatial embedded database : build

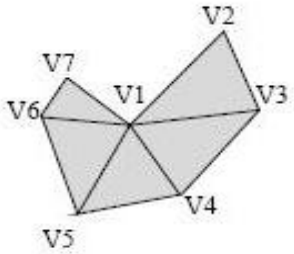


- Coverage: 64 countries
- Road coverage in North America: 13.6 million km
- Numbers of houses covered in North America: 300 million
- Road coverage in Europe: 8.5 million km
- Numbers of houses covered in Europe: 360 million
- Points of interest worldwide: 20 million
- Update: up to 4 times per year

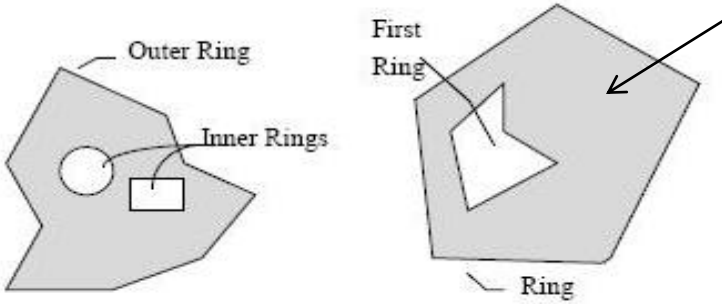
Spatial embedded database : content



Triangle Strip



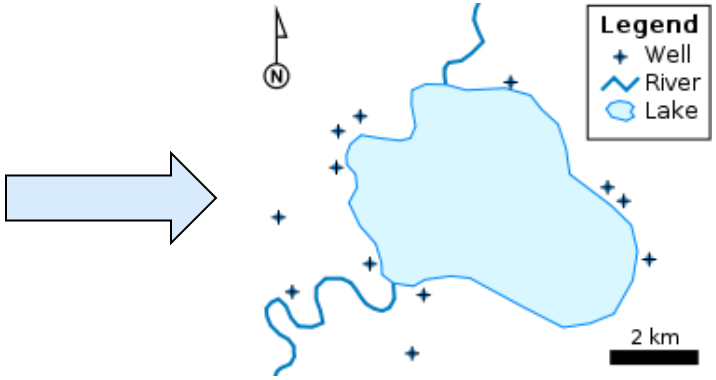
Triangle Fan



Value	Shape Type
0	Null Shape
1	Point
3	PolyLine
5	Polygon
8	MultiPoint
11	PointZ
13	PolyLineZ
15	PolygonZ
18	MultiPointZ
21	PointM
23	PolyLineM
25	PolygonM
28	MultiPointM
31	MultiPatch

A vector map, with points, polylines and polygons

```
PolygonZ
{
    Double[4]          Box          // Bounding Box
    Integer            NumParts     // Number of Parts
    Integer            NumPoints    // Total Number of Points
    Integer[NumParts]  Parts        // Index to First Point in Part
    Point[NumPoints]   Points       // Points for All Parts
    Double[2]          Z Range      // Bounding Z Range
    Double[NumPoints]  Z Array      // Z Values for All Points
    Double[2]          M Range      // Bounding Measure Range
    Double[NumPoints]  M Array      // Measures
}
```



Spatial embedded database : use

Database Spatial Search Index

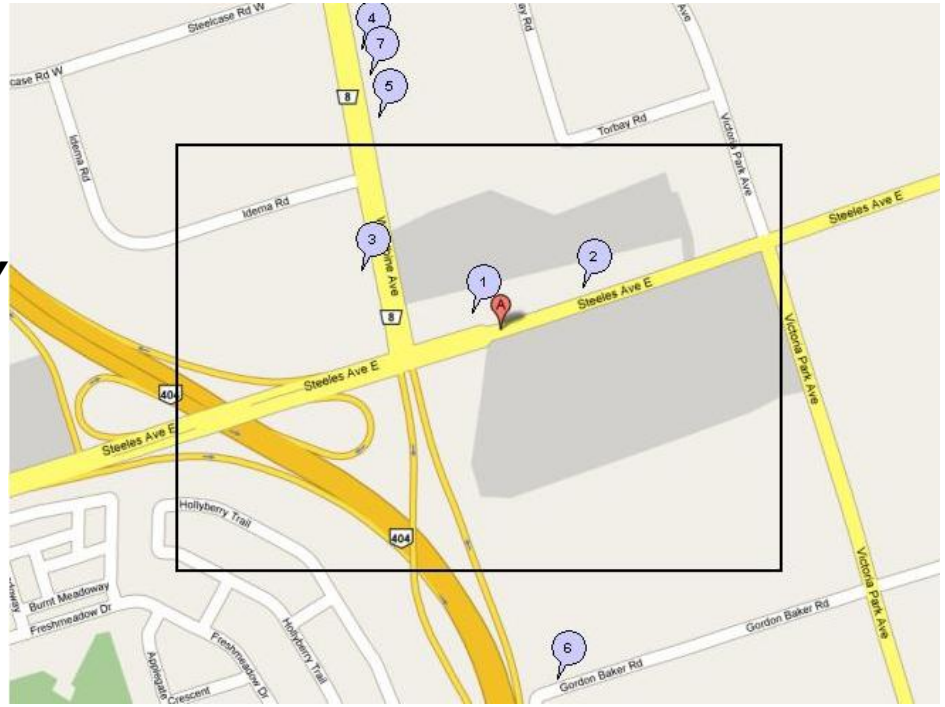
Latitude: 43 degrees 49 minutes 09.51 seconds North
Longitude: 79 degrees 20 minutes 53.41 seconds West

The latitude 43 degrees 49 minutes 09.51 seconds North is converted to:
43 * 360000 + 49 * 6000 + 9.51*100 = **15774951 (x)**
The longitude 79 degrees 20 minutes 53.41 seconds West is converted to:
79 * 360000 + 20 * 6000 + 53.41*100 = **28565341 (y)**

EMPRESS SOFTWARE INC.

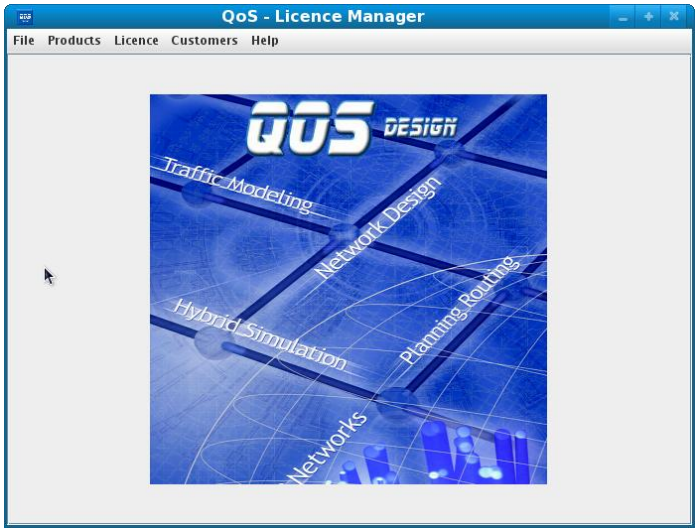
Points of Interest inside defined rectangle XY

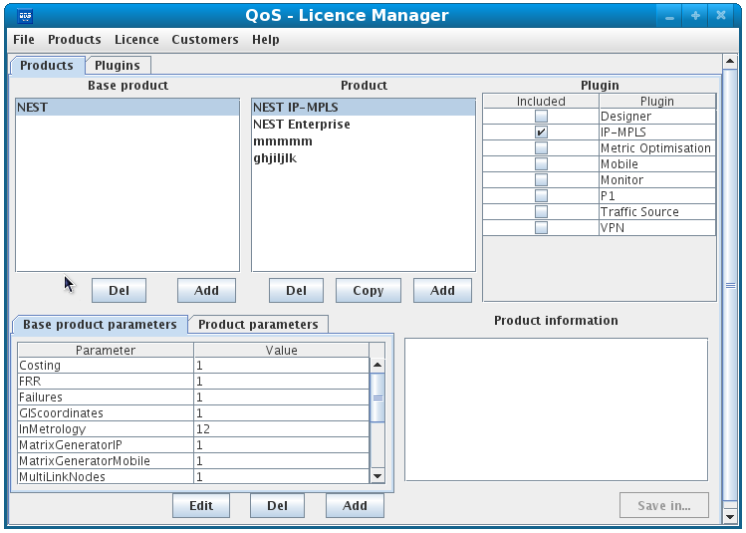
id	name	X coordinate	Y coordinate
1	Ten23	28565120	15773195
2	Buffet City	28563997	15773487
3	Golden Griddle	28565768	15773777



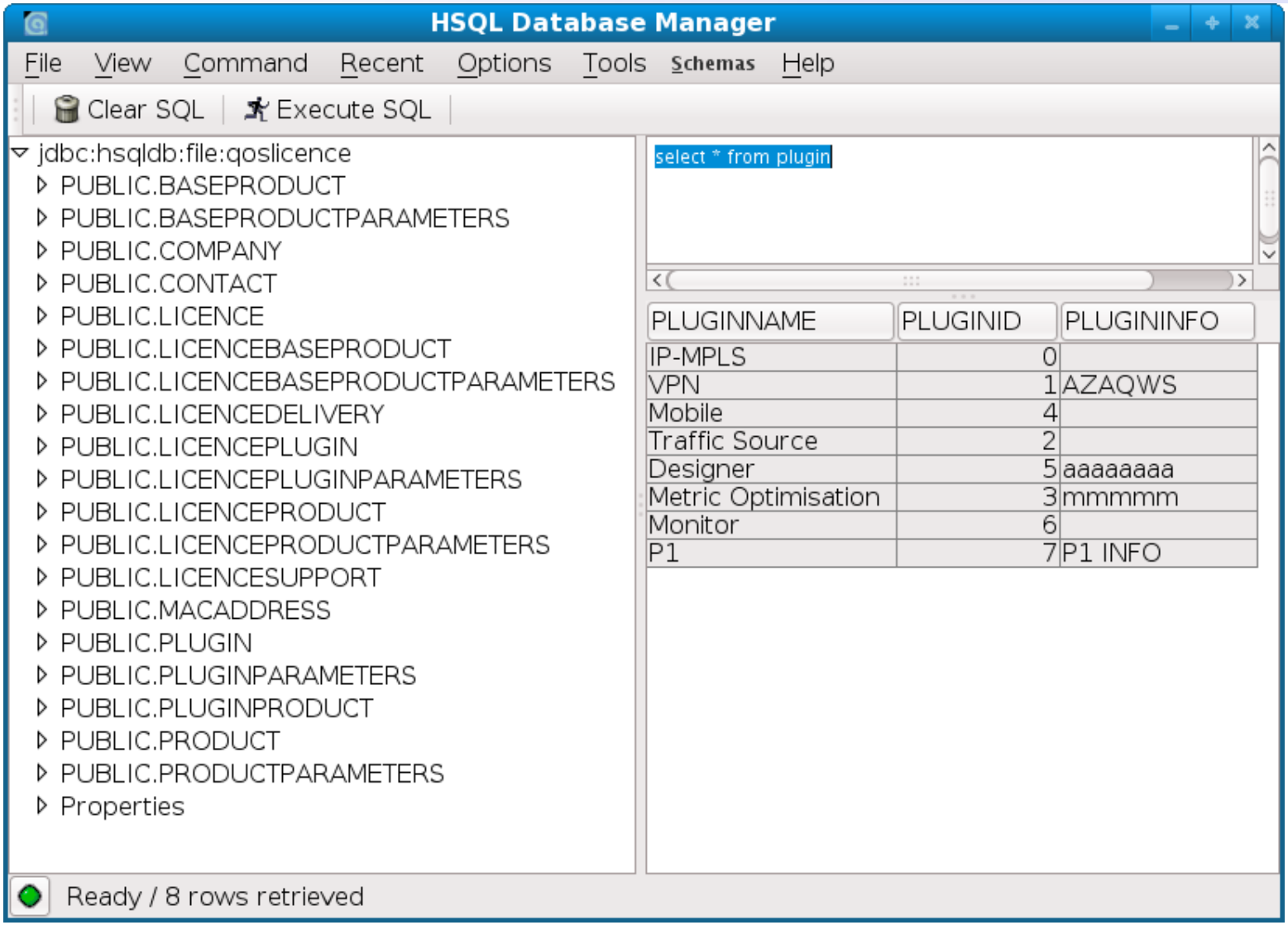
Small embedded database : HSQLDB

- HyperSQL DataBaseSQL relational database engine written in Java. http://hsqldb.org
- It has a JDBC driver and supports nearly full ANSI-92 SQL
- **In-memory** and **disk-based** tables and supports **embedded** and **server** modes.
- It includes tools such as a command line SQL tool and GUI query tools.
- Can switch from differents database : Oracle, Mysql …





Small embedded database : HSQLDB




```
java -cp "../lib/hsqldb.jar" org.hsqldb.util.DatabaseManagerSwing
```

Small embedded database : HSQLDB

databaseName.script

```
CREATE SCHEMA PUBLIC AUTHORIZATION DBA

CREATE TEXT TABLE BASEPRODUCT(PRODUCTNAME VARCHAR(100) NOT
NULL,PRODUCTID INTEGER,PRODUCTINFO VARCHAR(4000) NOT
NULL,CONSTRAINT SYS_CT_46 UNIQUE(PRODUCTID))

CREATE INDEX INDEXBASEPRODUCT ON BASEPRODUCT(PRODUCTID)

SET TABLE BASEPRODUCT SOURCE "baseproduct“

CREATE USER QOS PASSWORD “xxx"
GRANT DBA TO QOS
SET WRITE_DELAY 10
```

Small embedded database : HSQLDB

databaseName.properties

```
#HSQL Database Engine 1.8.0.7
#Sat Jan 09 16:17:36 CET 2010
hsqldb.script_format=0
runtime.gc_interval=0
sql.enforce_strict_size=false
hsqldb.cache_size_scale=8
readonly=false
hsqldb.nio_data_file=true
hsqldb.cache_scale=14
version=1.8.0
hsqldb.default_table_type=text
hsqldb.cache_file_scale=1
hsqldb.log_size=200
modified=yes
hsqldb.cache_version=1.7.0
hsqldb.original_version=1.8.0
hsqldb.compatible_version=1.8.0
```

Small embedded database : HSQLDB

How to use HSQLDB ?

1) Load driver :

```
Class.forName("org.hsqldb.jdbcDriver").newInstance();
DataBaseAccess  dataAccess = new DataBaseAccess(this);
```

2) Connect :

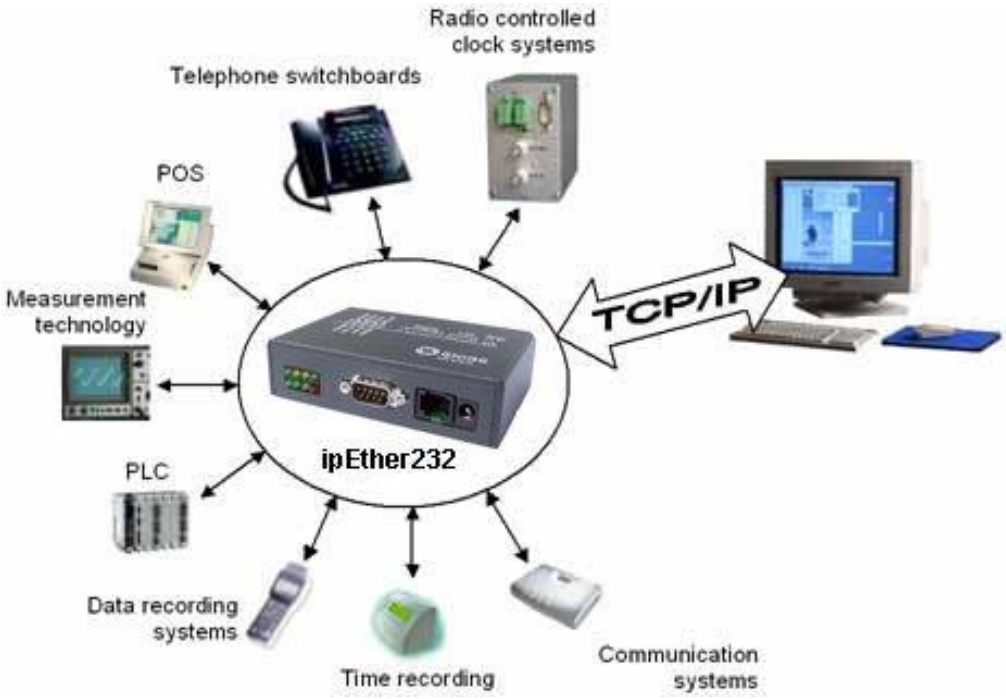
```
Connection  connexion = DriverManager.getConnection("jdbc:hsqldb:file:"
+ path + databaseName, login, password);
```

3) Query :

```
Statement st = connexion.createStatement();
ResultSet rs = executeQuery(expression); ;
return rs;
```

Conclusion

- The database must be adapted to the constraints of embedded systems:
- Critical / noncritical
- High performance / performance standard
- Volume of data to store
- Support: Memory / Disk
- Type of contents
- Specific : « Homemade » database



Useful links

- A database technology for high-performance applications.
- Wikipedia Embedded database
- eXtremeDB
- Hsqldb
- Database Spatial
- Real time database
- Comparison of relational database management systems
- Referential integrity
- Shapefile
- Tele Atlas MultiNet
- Kernel (computing)

what does that plane embed ?

11111011010 - National Network of Software Developer -11111011010

Mailing list : DEVLOG@cnrs.services.fr
Mailing list comity : Devlog-contact@cnrs.services.fr