Relational Database Schema Evolution

An Industrial Case Study

Julien Delplanque, Anne Etien, Nicolas Anquetil and Olivier Auverlot

{name}.{surname}@inria.fr

Université de Lille

inventeurs du monde numérique

RMod

https://rmod.inria.fr/web/

Roadmap

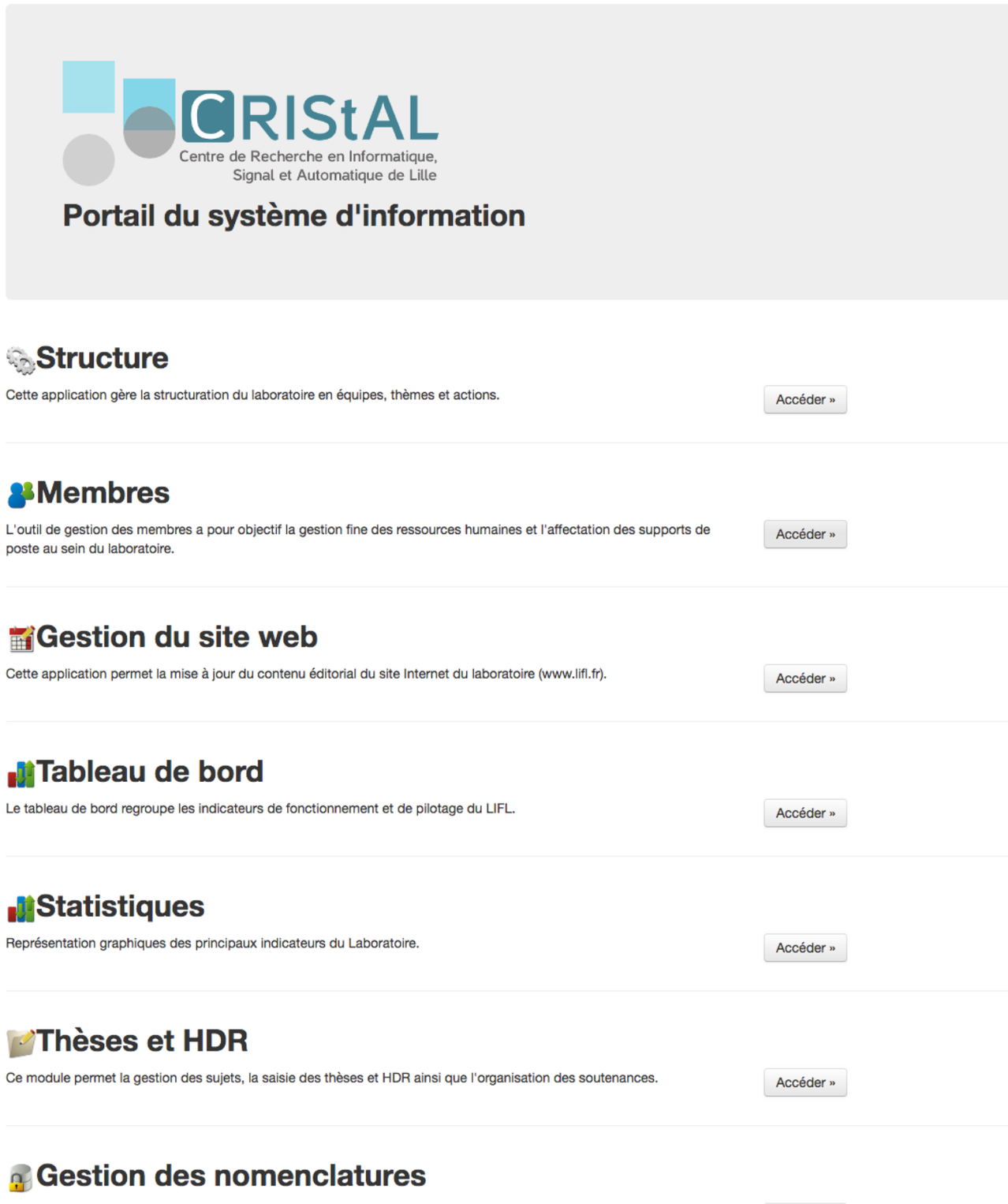
1. **AppSI database overview**
2. An evolution of AppSI
3. Experimental setup
4. Qualitative and Quantitative analyses
5. Observed problems

AppSI database overview

- Members, teams, funding support, etc. management in laboratories of our university;

- Used by software systems written in different programming languages;

- Used in multiple laboratories at the university.



Structural entities

- Tables: 95
- Columns: 515
- PK constraints: 93
- FK constraints: 125
- Other constraints: 118

Behavioural entities

- Views: 62
- Triggers: 20
- Functions: 86

} Behaviour integrated inside the DB

Roadmap

1. AppSI database overview
2. **An evolution of AppSI**
3. Experimental setup
4. Qualitative and Quantitative analyses
5. Observed problems

An evolution of AppSI

person
id : serial (PK)
uid : varchar
email : varchar
...

Before the modification,

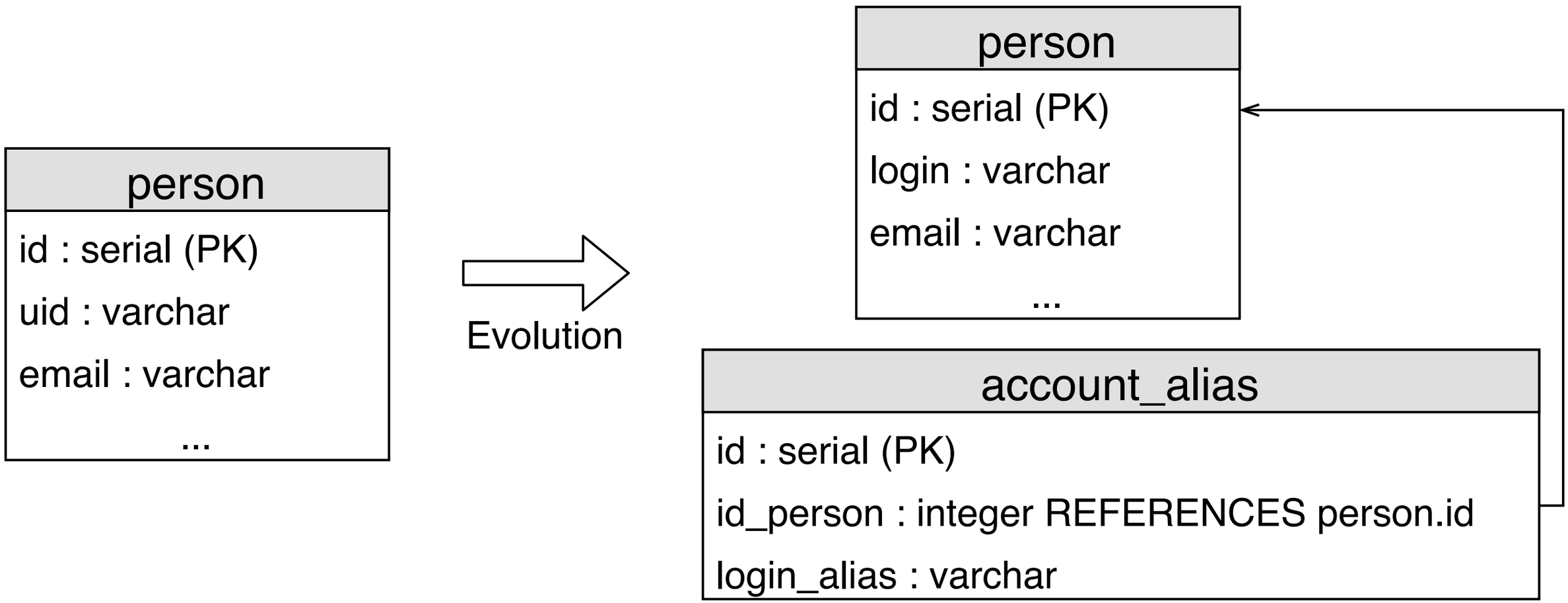
- It has a primary key (*id*) of type serial
- It stores the LDAP identifiers in the *uid* column

New LDAP schema

- Users may have multiple identifiers
- *uid* attribute has been renamed into *login*

Consequently, *uid* column of *person* needs to be renamed into *login*.

An evolution of AppSI



After the modification,

- Allow users to have multiple identifiers ➡ *account_alias* table is created
- Gather all the secondary identifiers of a *person* ➡ *login_alias* column

An evolution of AppSI

Before	After
A person has a **single** identifier	A person may have **multiple** identifier
The *id* of a person can be retrieved **directly**	The *id* of a person needs to be **computed**

The architect will:

- Use a **roadmap** to keep track of his progress;
  - **updated during the evolution**;
- Materialize the evolution as a **SQL patch**;
  - **implementation of the roadmap**.

Roadmap

1. AppSI database overview
2. An evolution of AppSI
3. **Experimental setup**
4. Qualitative and Quantitative analyses
5. Observed problems

Experimental setup

- The architect of AppSI got his screen recorded during 3 development session.

Video #	Total time	Entries #
1	1:55:30	100
2	1:19:17	114
3	0:52:41	96

1. Transcribe the videos into a list of **entries** corresponding to changes in the screen display;
2. Abstract these entries into distinct **actions**;
3. Group these actions into **activities**;
4. **Formalize** the intuitive process followed by the architect as an **activity diagram**;
5. **Discuss** analysis **with the architect**; and
6. **Analyse** quantitatively data extracted from steps 2, 3 and 4;

Decomposing the evolution

- Videos were transcribed as **entries**
  - ’00:07:24,00:07:39,Go back on the trigger and remove it without error.’
- Entries are generalised as **actions**
  - ‘Observe patch’
  - ‘Execute DDL query from IDE’
  - ‘Inactivity’
  - ‘Other’
- Actions are grouped into **activities**
  - ‘Implement changes into queries’
  - ‘Execute queries in a transaction and commit’

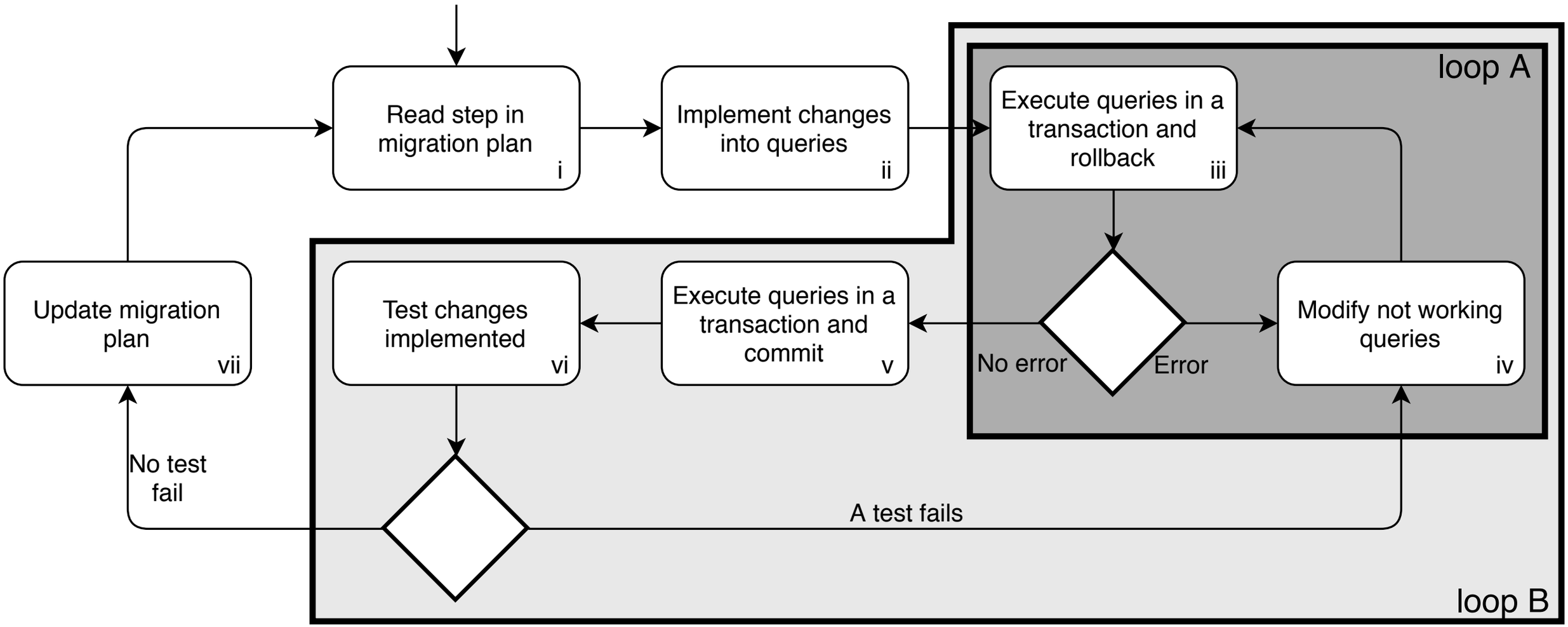
Roadmap

1. AppSI database overview
2. An evolution of AppSI
3. Experimental setup
4. **Qualitative and Quantitative analyses**
5. Observed problems

Analysis

- More than **4h of videos** transcribed/analysed in ~ 30-40h by 2 people
- **312 entries** extracted
- **18 actions** identified
- **7 activities** identified
- **Validation** of the analysis by a second person

An informal process



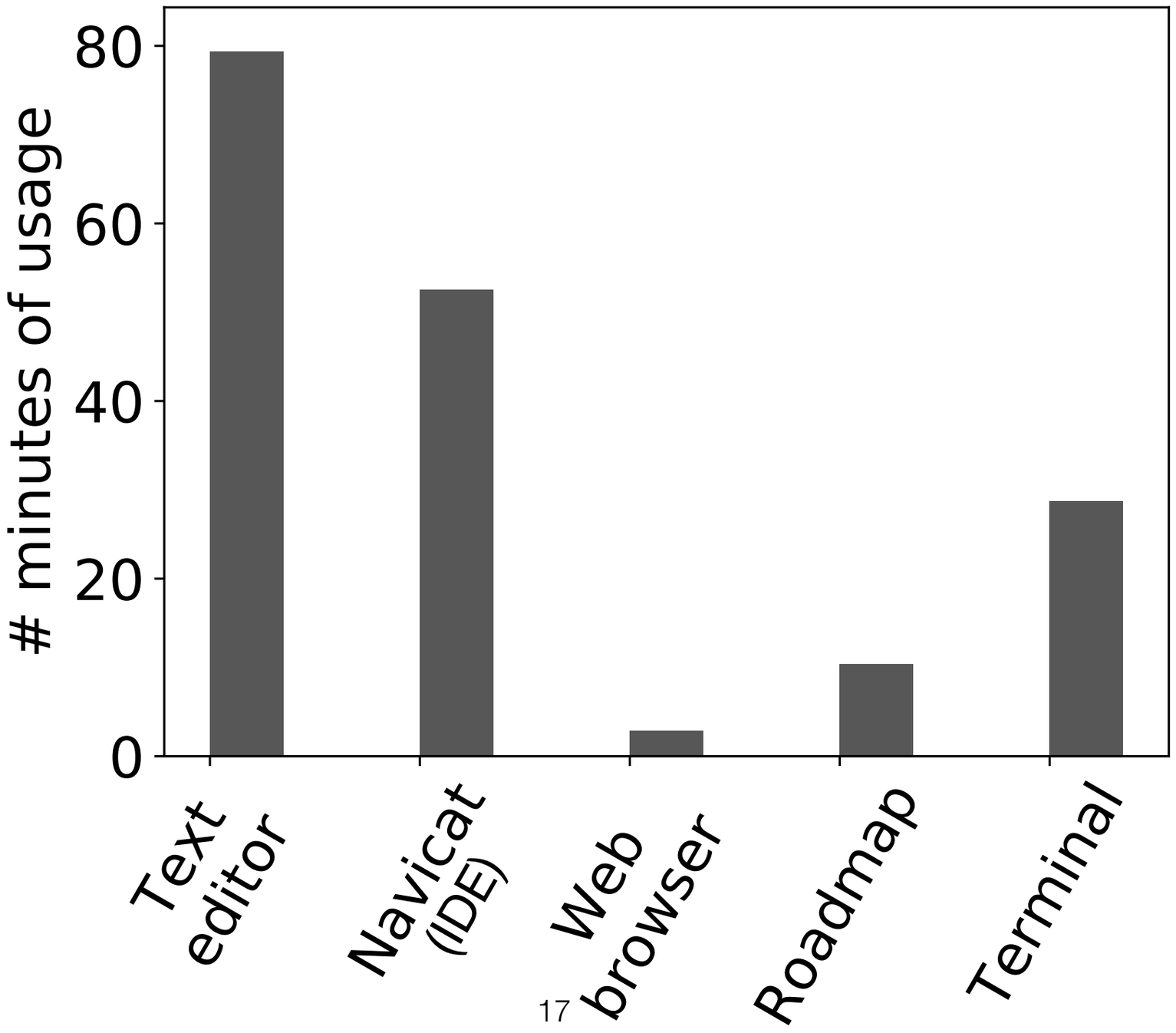
- **Loop A:** resolution of syntax errors and execution errors (e.g. reference to nonexistent entities).
- **Loop B**: resolution of semantic errors.
- **Main Loop**: complete implementation of a feature (might include multiple iterations on loop A and/or loop B).

Quantitative Analysis of Architect's Activities

3 interesting cases:

1. Iteration **without error**: **3/9** main loop iterations
2. Multiple iterations on **sub-loop A**: up to **6** iterations
3. Multiple iterations on **sub-loop B**: up to **3** iterations

Quantitative Analysis of Architect's Activities



Roadmap

1. AppSI database overview
2. An evolution of AppSI
3. Experimental setup
4. Qualitative and Quantitative analyses
5. **Observed problems**

Observed problems

A. Analysing and Visualising Dependencies Between Database's Entities

B. Evaluating the Impact of a Modification on the Database

C. Managing Co-evolution of Multiple Instances of a Database Schema

D. Testing Database's Functionalities

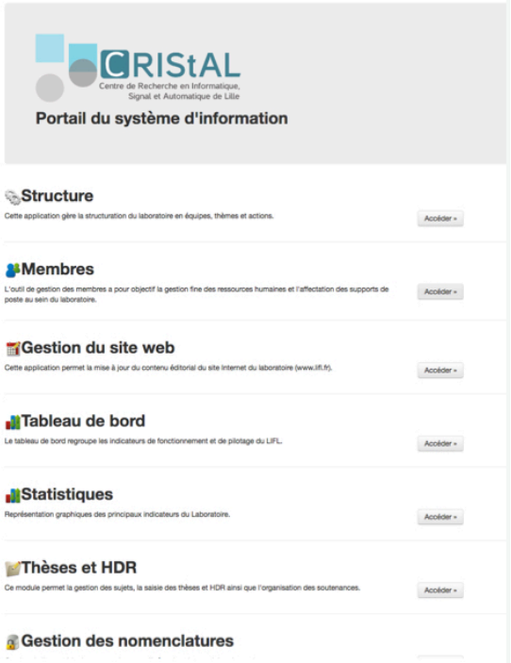
E. Synchronisation of IDE's Internal State according to Database Architect's Actions

F. No Integrated Solution

Conclusion

AppSI database overview

- Members, teams, funding support, etc. management in laboratories of our university;
- Used by software systems written in different programming languages;
- Used in multiple laboratories at the university.



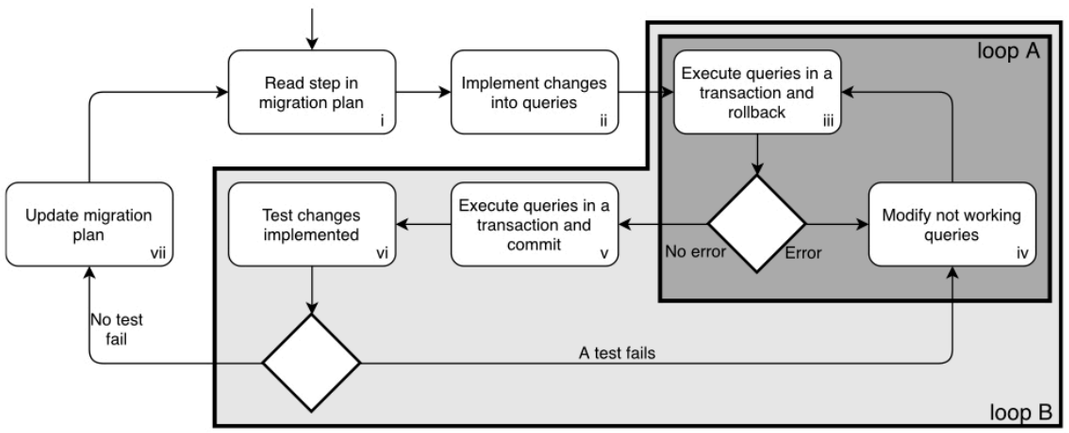
An evolution of AppSI

Before	After
A person has a **single** identifier	A person may have **multiple** identifier
The *id* of a person can be retrieved **directly**	The *id* of a person needs to be **computed**

The architect will:

- Use a **roadmap** to keep track of his progress;
  - **updated during the evolution**;
- Materialize the evolution as a **SQL patch**;
  - **implementation of the roadmap**.

An informal process



- **Loop A:** resolution of syntax errors and execution errors (e.g. reference to nonexistent entities).
- **Loop B**: resolution of semantic errors.
- **Main Loop**: complete implementation of a feature (might include multiple iterations on loop A and/or loop B).

Observed problems

A. Analysing and Visualising Dependencies Between Database's Entities

B. Evaluating the Impact of a Modification on the Database

C. Managing Co-evolution of Multiple Instances of a Database Schema

D. Testing Database's Functionalities

E. Synchronisation of IDE's Internal State according to Database Architect's Actions

F. No Integrated Solution

Future work

- Apply software engineering techniques to database management

- Adopt a model-based analysis to identify dependencies between entities

- Recommendation system

Decomposing the evolution

Action #	Description
0	Other.
1	Synchronise development database
2	Observe patch.
3	Observe DB entities.
4	Search in database schema dump.
5	Syntax search in database schema
6	Execute SELECT query from IDE.
7	Execute INSERT/UPDATE/DELETE
8	Execute DDL query from IDE's UI.
9	Execute DDL query from IDE.
10	Modify patch.
11	Change application verification.
12	Update evolution roadmap.
13	Check PostgreSQL documentation.
14	Check evolution roadmap.
15	Modify source code in query builder.
16	Run unit tests written in an external
17	Inactivity.

Actions

Activity	Actions
(i) Read step in migration plan	14
(ii) Implement changes into queries	2, 3, 4, 5, 10 or 13
(iii) Execute queries in a transaction and rollback	9
(iv) Modify not working queries	2, 3, 4, 5, 8, 10, 13 or 15
(v) Execute queries in a transaction and commit	9
(vi) Tests changes implemented	3, 6, 7, 11 or 16
(vii) Update migration plan	12

Activities

Quantitative Analysis of Architect's Activities

ID	Duration (min)	# Actions
1	62.50	100
2	22.73	27
3	4.13	8
4	9.45	22
5	18.33	41
6	13.20	8
7	8.35	36
8	16.01	23
9	28.28	41

Main loops

Main loop id	Type	Duration (min)	# Actions
1	A	2.46	5
1	A	0.65	2
1	A	2.81	2
1	A	6.68	14
1	A	4.36	10
1	B	0.38	3
2	B	1.00	3
4	B	0.48	2
4	B	0.96	1
4	B	1.01	4
4	A	3.48	7
5	B	0.96	3
5	A	1.60	7
7	A	5.41	19
9	A	5.18	9

Sub-loops

- **Loop A:** concerns the resolution of syntax errors and errors raised because of reference to nonexistent entities.
- **Loop B**: concerns the resolution of semantic errors.
- **Main Loop**: concerns a complete implementation of a feature. Such loop might include multiple iterations on loop A and/or loop B.

Quantitative Analysis of Architect's Activities