Recommendations

for Evolving Relational Databases

Julien Delplanque, Anne Etien, Nicolas Anquetil and Stéphane Ducasse

{firstname}.{lastname}@inria.fr

Université de Lille

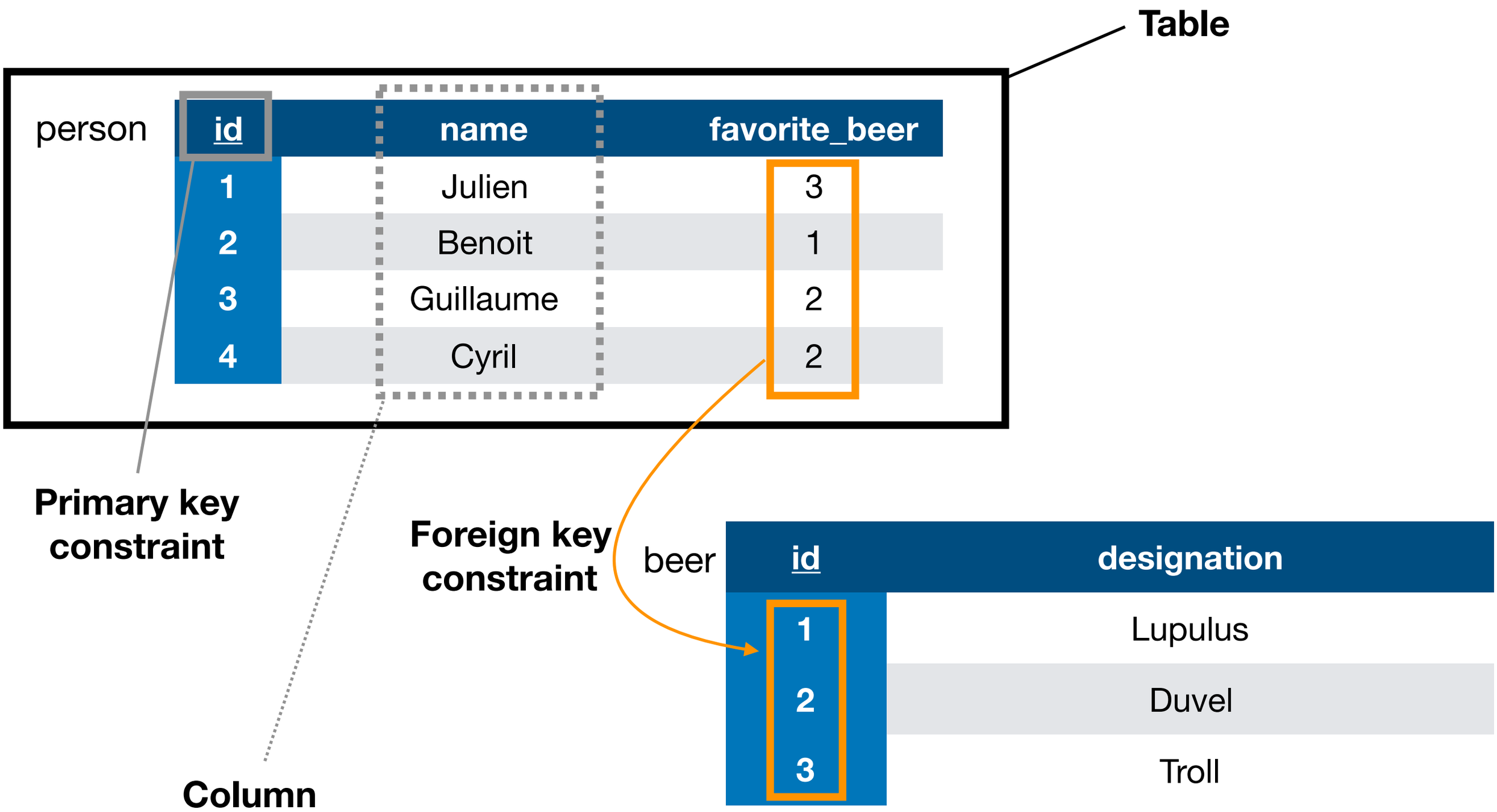
Inria
inventeurs du monde numérique

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

Overview

1. Reminder about relational databases
2. Problems during relational database evolution
3. Recommendations for Evolving Relational Databases
4. Conclusions and future work

RDBMS - Structural Entities



RDBMS - Behavioural entities

- Views: Named SELECT queries stored in the database.
- Stored procedures: Functions, written in a programming language, implementing arbitrary computation.
- Triggers: Entity listening to events happening on a table and reacting to them.
- …

Problems during DB evolution

1. Relational database management systems (RDBMS) do not allow schema inconsistencies.

2. Stored procedure bodies are not meta-described.

Examples

Example: remove person.name

person

id	~~name~~	favorite_beer
1	~~Julien~~	3
2	~~Benoit~~	1
3	~~Guillaume~~	2
4	~~Cyril~~	2

beer

id	designation
1	Lupulus
2	Duvel
3	Troll

Example: remove person.name

person

id	~~name~~	favorite_beer
1	~~Julien~~	3
2	~~Benoit~~	1
3	~~Guillaume~~	2
4	~~Cyril~~	2

Change applied by RDBMS

beer

id	designation
1	Lupulus
2	Duvel
3	Troll

Example: remove beer.id

person

id	name	favorite_beer
1	Julien	3
2	Benoit	1
3	Guillaume	2
4	Cyril	2

beer

~~id~~	designation
~~1~~	Lupulus
~~2~~	Duvel
~~3~~	Troll

Example: remove beer.id

person

id	name	favorite_beer
1	Julien	3
2	Benoit	1
3	Guillaume	2
4	Cyril	2

beer

id	designation
1	Lupulus
2	Duvel
3	Troll

Forbidden by the RDBMS!

Example: remove column beer.designation referenced in view

person

id	name	favorite_beer
1	Julien	3
2	Benoit	1
3	Guillaume	2
4	Cyril	2

beer

id	~~designation~~
1	~~Lupulus~~
2	~~Duvel~~
3	~~Troll~~

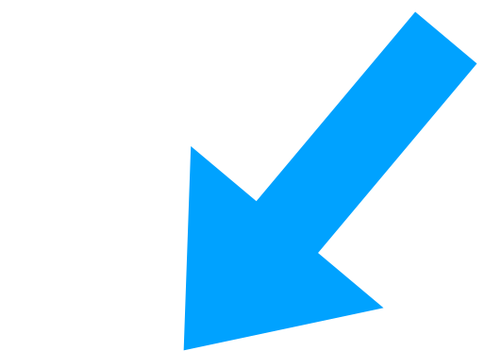
VIEW person_to_favorite_designation

```
SELECT id, name, designation
FROM person, beer
WHERE person.favorite_beer = beer.id;
```

Example: remove column beer.designation referenced in view

VIEW person_to_favorite_designation

```
SELECT id, name, designation
FROM person, beer
WHERE person.favorite_beer = beer.id;
```



Refuse change

VIEW person_to_favorite_designation

```
SELECT id, name, designation
FROM person, beer
WHERE person.favorite_beer = beer.id;
```

⋮

Continues in cascade

Example: remove column beer.designation referenced in view

VIEW person_to_favorite_designation

```
SELECT id, name, designation
FROM person, beer
WHERE person.favorite_beer = beer.id;
```

Not convenient behaviour

Refuse change

VIEW person_to_favorite_designation

```
SELECT id, name, designation
FROM person, beer
WHERE person.favorite_beer = beer.id;
```

⋮

Continues in cascade

Example: remove person.name

person

id	name	favorite_beer
1	Julien	3
2	Benoit	1
3	Guillaume	2
4	Cyril	2

beer

id	designation
1	Lupulus
2	Duvel
3	Troll

STOR. PROC. id_of_troll_lovers

```
RETURN SELECT person.id
FROM person, beer
WHERE person.favorite_beer = beer.id
   AND beer.designation = 'Troll';
```

Example: remove person.name

person

id	name	favorite_beer
1	Julien	3
2	Benoit	1
3	Guillaume	2
4	Cyril	2

Unseen by the RDBMS!

beer

id	designation
1	Lupulus
2	Duvel
3	Troll

STOR. PROC. id_of_troll_lovers

```
RETURN SELECT person.id
FROM person, beer
WHERE person.favorite_beer = beer.id
   AND beer.designation = 'Troll';
```

Recommendations for Evolving Relational Databases

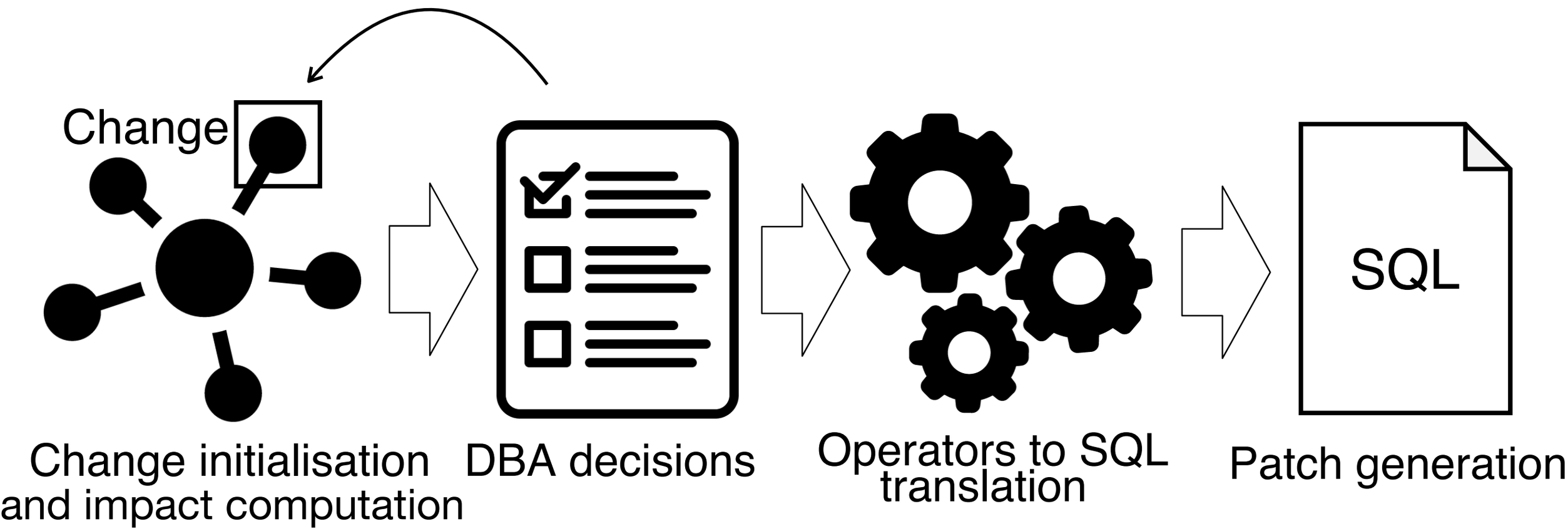
Refactoring v.s. Change

	Refactoring	Change
Code evolution	✅	✅
Behaviour preserving	✅	❌
Modify software features	❌	✅

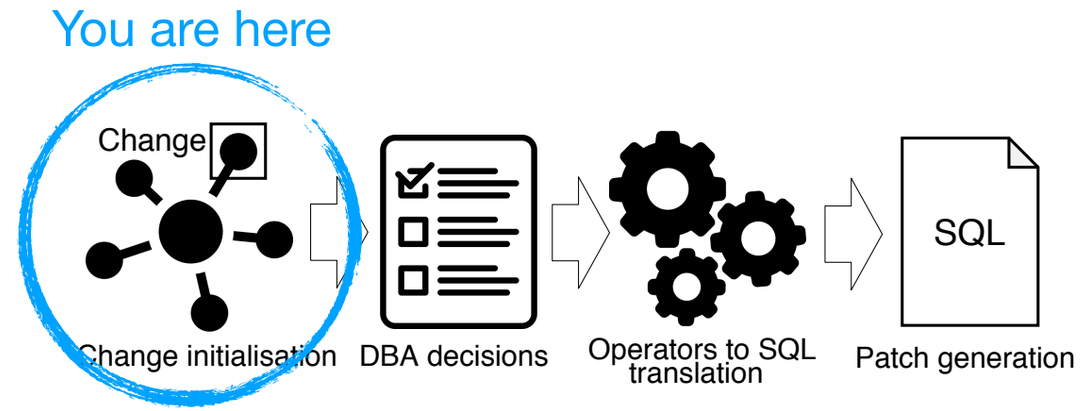
Refactoring v.s. Change

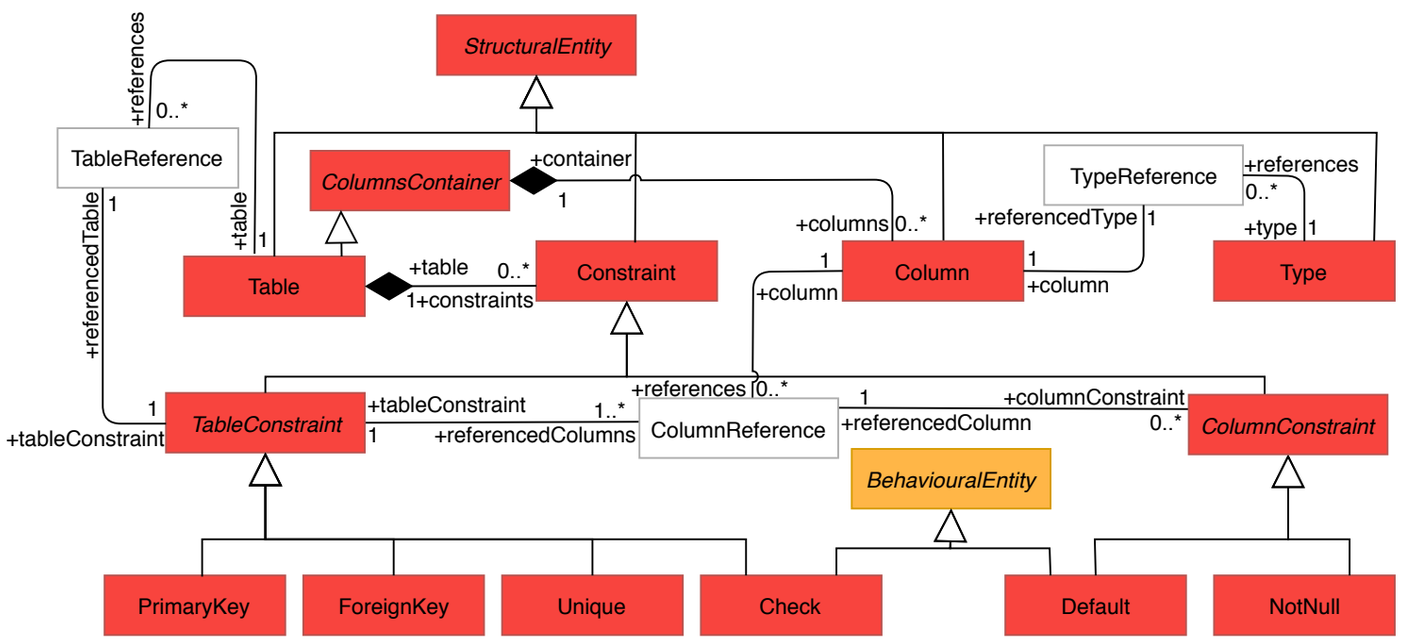
	Refactoring	Change
Code evolution	✅	✅
Behaviour preserving	✅	❌
Modify software features	❌	✅

Approach overview

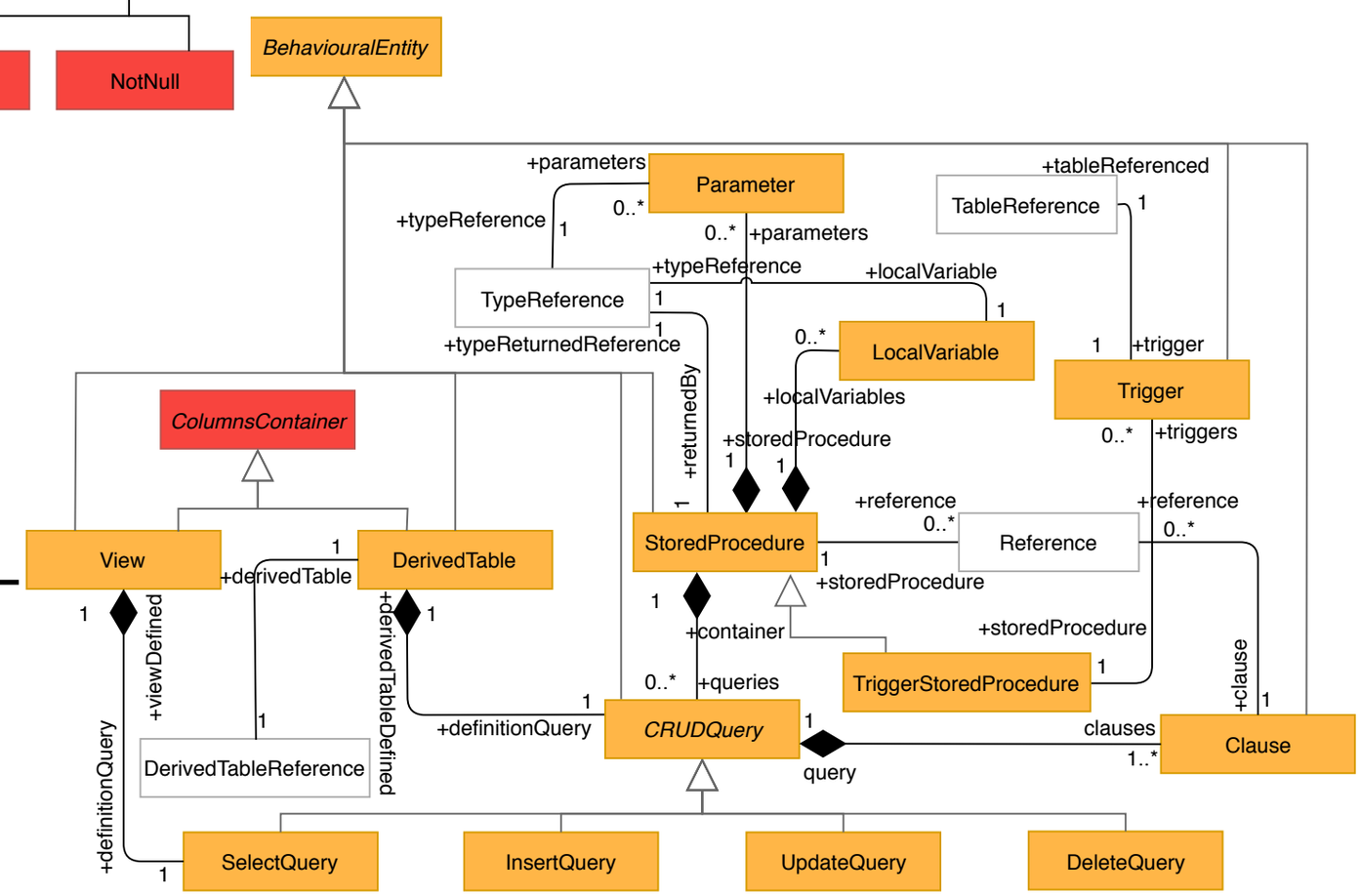


Meta-model



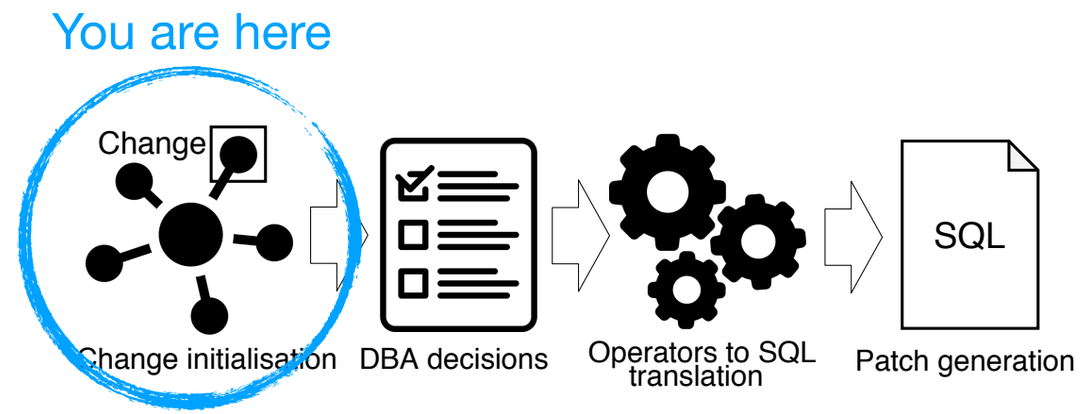


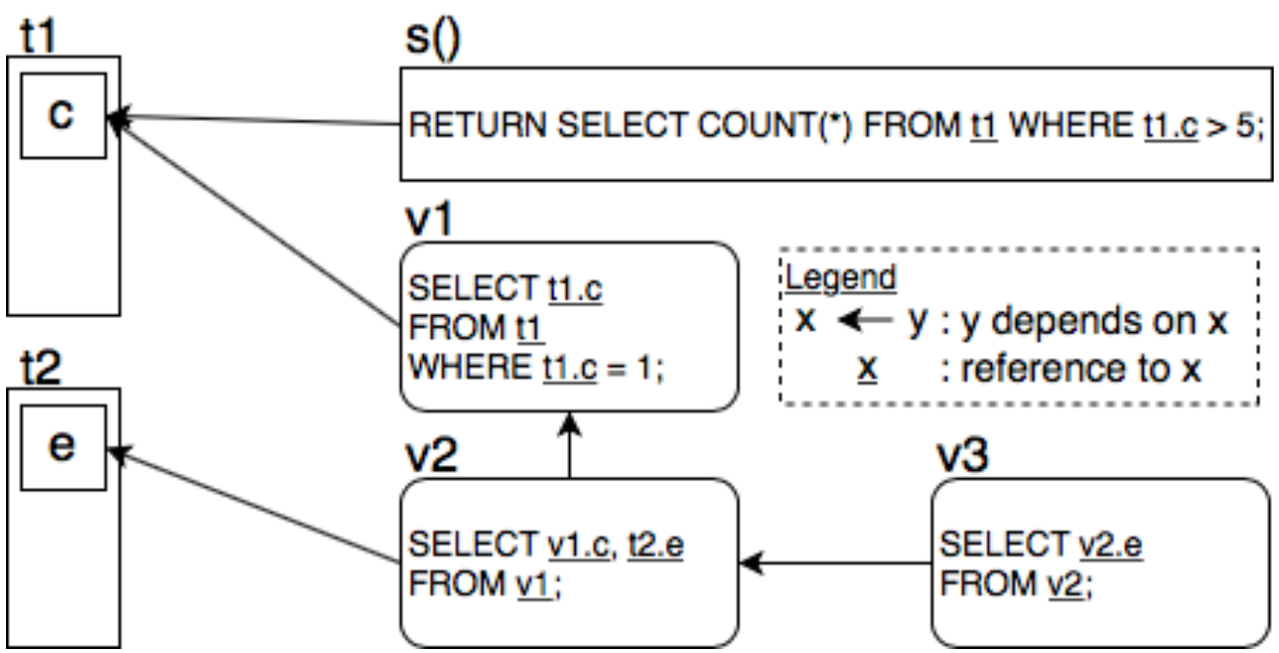
Structural entities



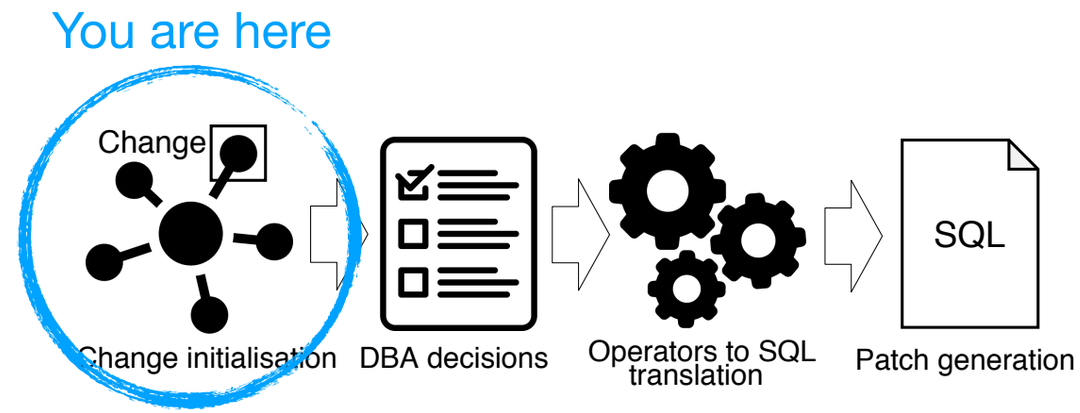
Behavioural entities

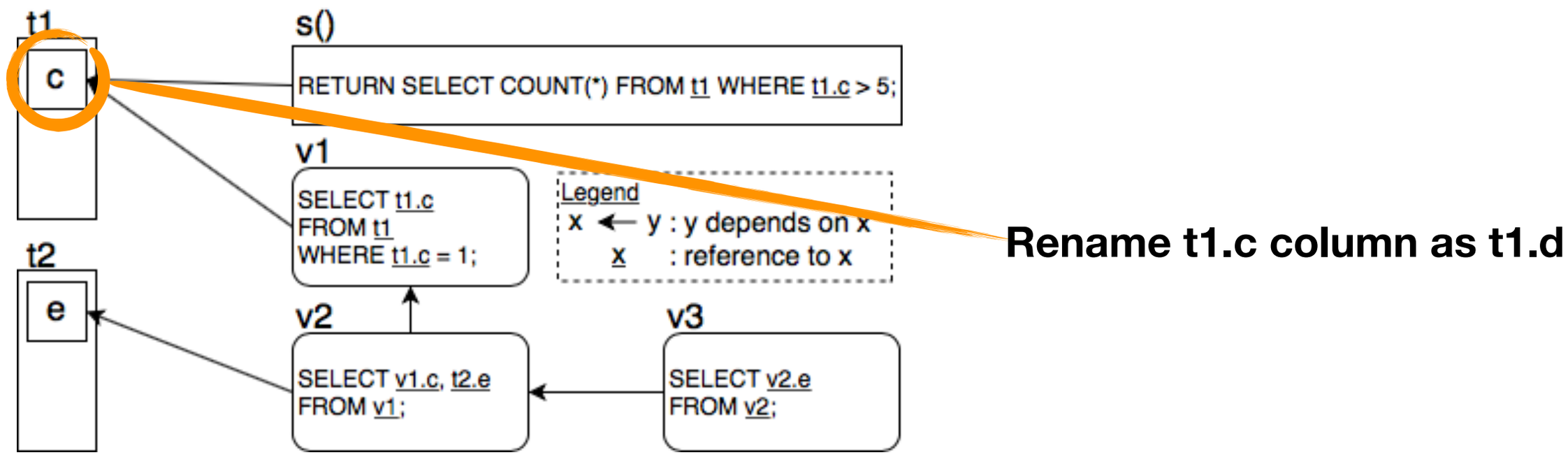
Model


You are here
Change
Change initialisation
DBA decisions
Operators to SQL translation
SQL
Patch generation

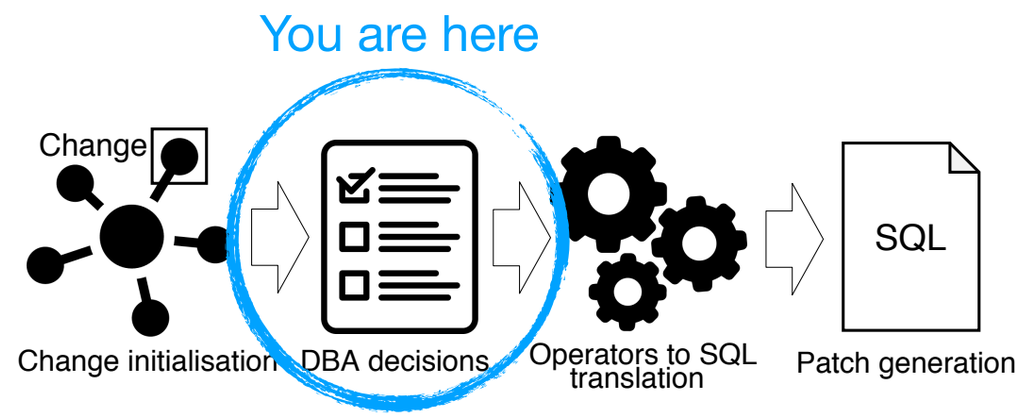

t1
c
t2
e
s()
RETURN SELECT COUNT(*) FROM t1 WHERE t1.c > 5;
v1
SELECT t1.c
FROM t1
WHERE t1.c = 1;
Legend
x ← y : y depends on x
x : reference to x
v2
SELECT v1.c, t2.e
FROM v1;
v3
SELECT v2.e
FROM v2;

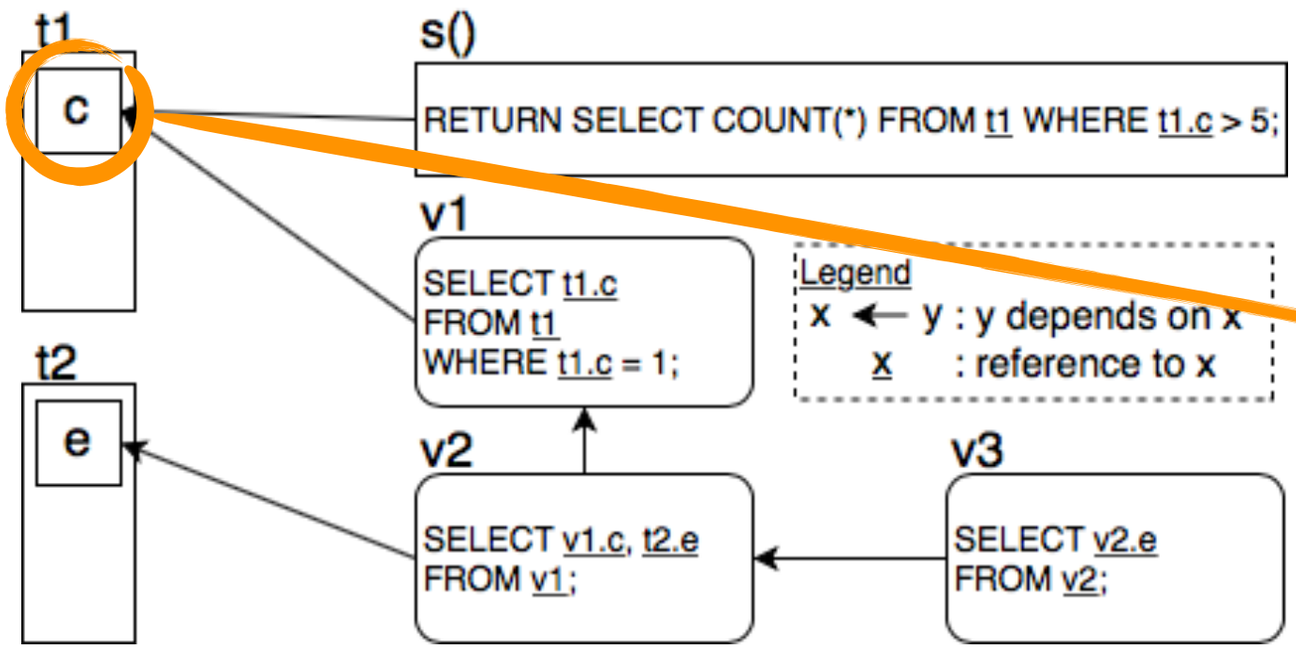
Model


You are here
Change
Change initialisation
DBA decisions
Operators to SQL translation
SQL
Patch generation

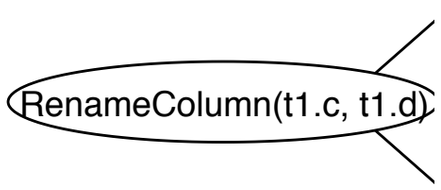

t1
c
t2
e
s()
RETURN SELECT COUNT(*) FROM t1 WHERE t1.c > 5;
v1
SELECT t1.c
FROM t1
WHERE t1.c = 1;
Legend
x ← y : y depends on x
x : reference to x
v2
SELECT v1.c, t2.e
FROM v1;
v3
SELECT v2.e
FROM v2;
Rename t1.c column as t1.d

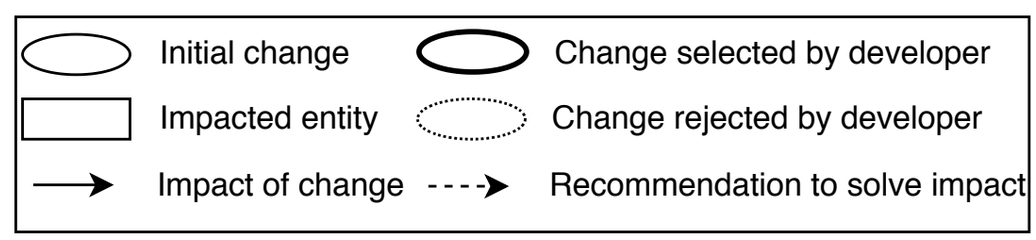
Impact computation & Recommendations selection



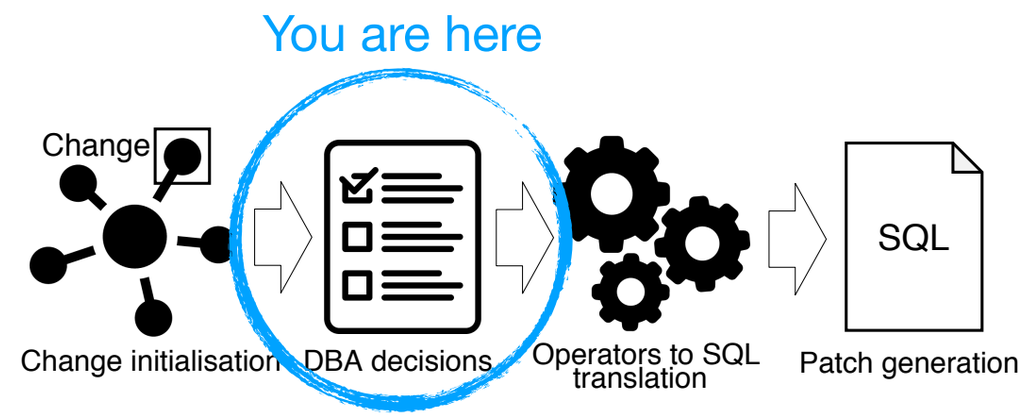


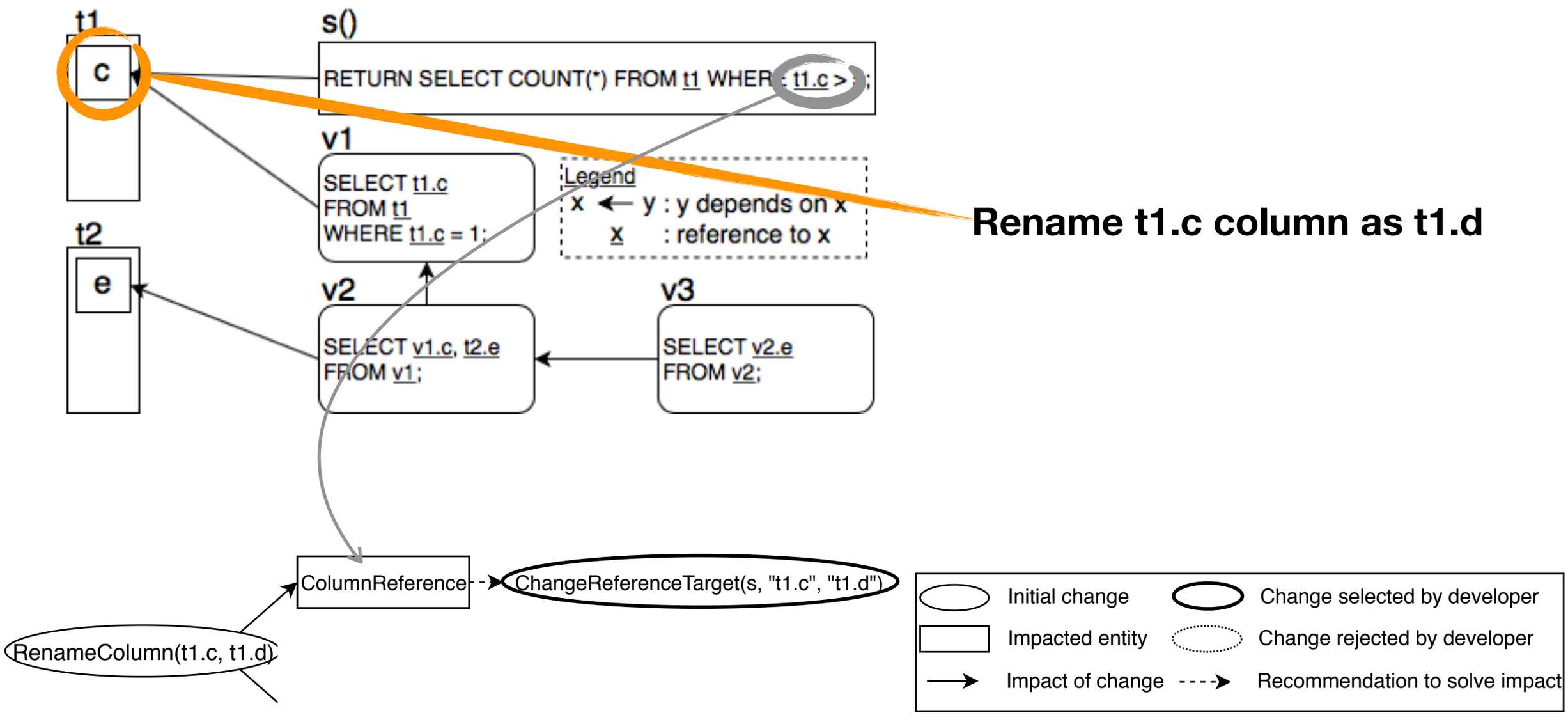
Rename t1.c column as t1.d



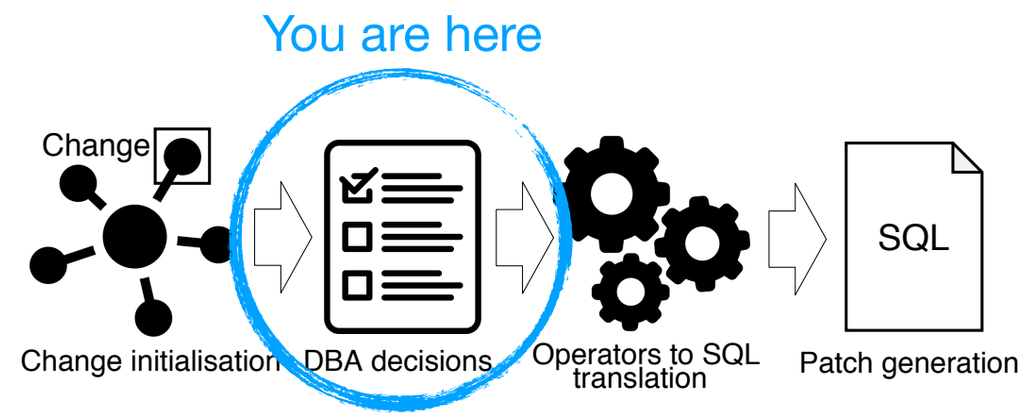


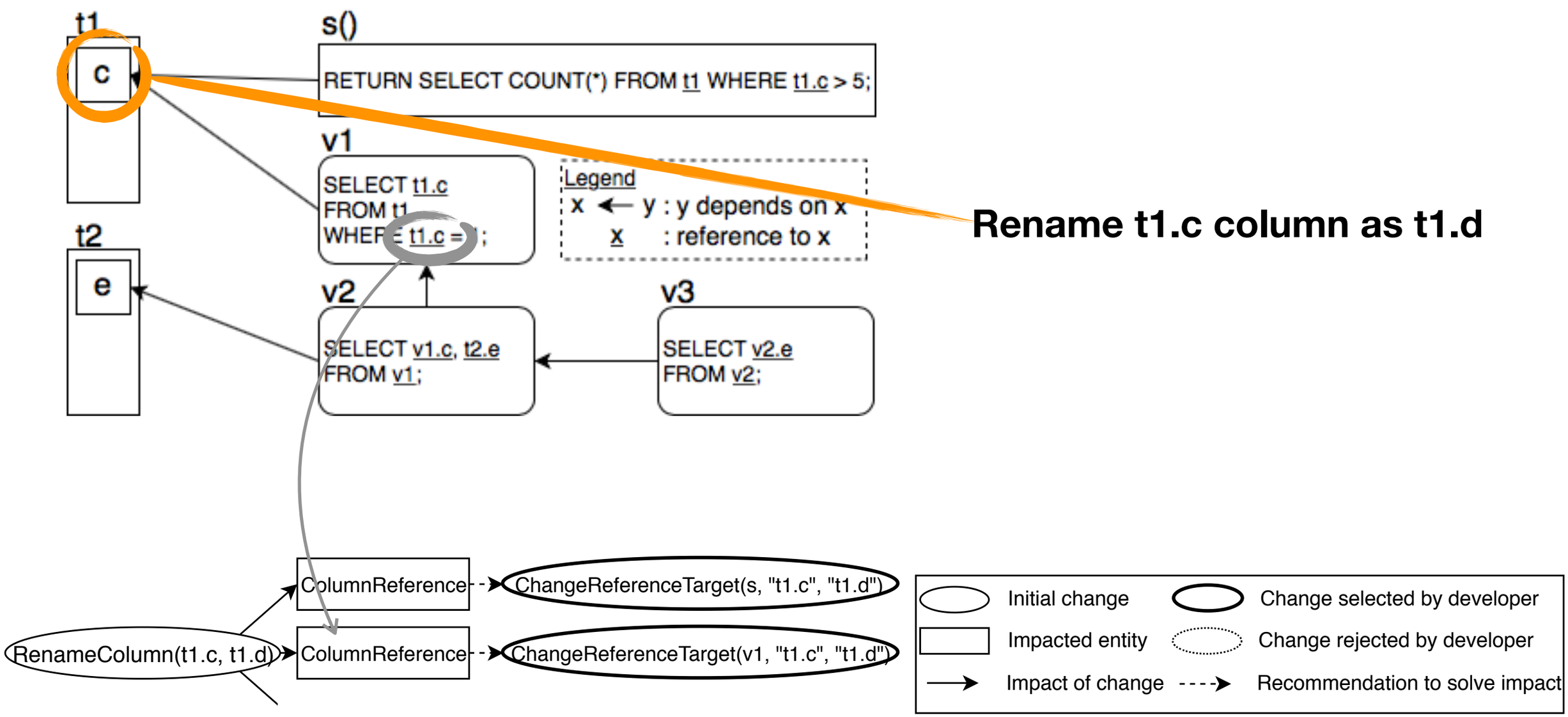
Impact computation & Recommendations selection



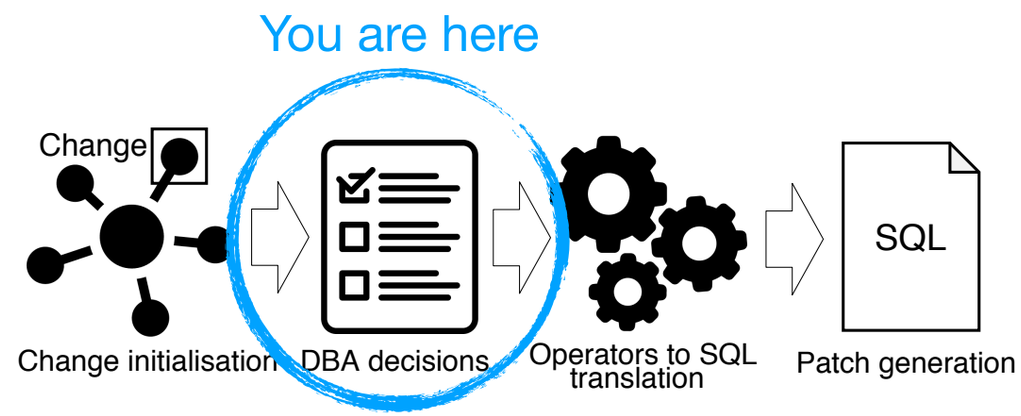


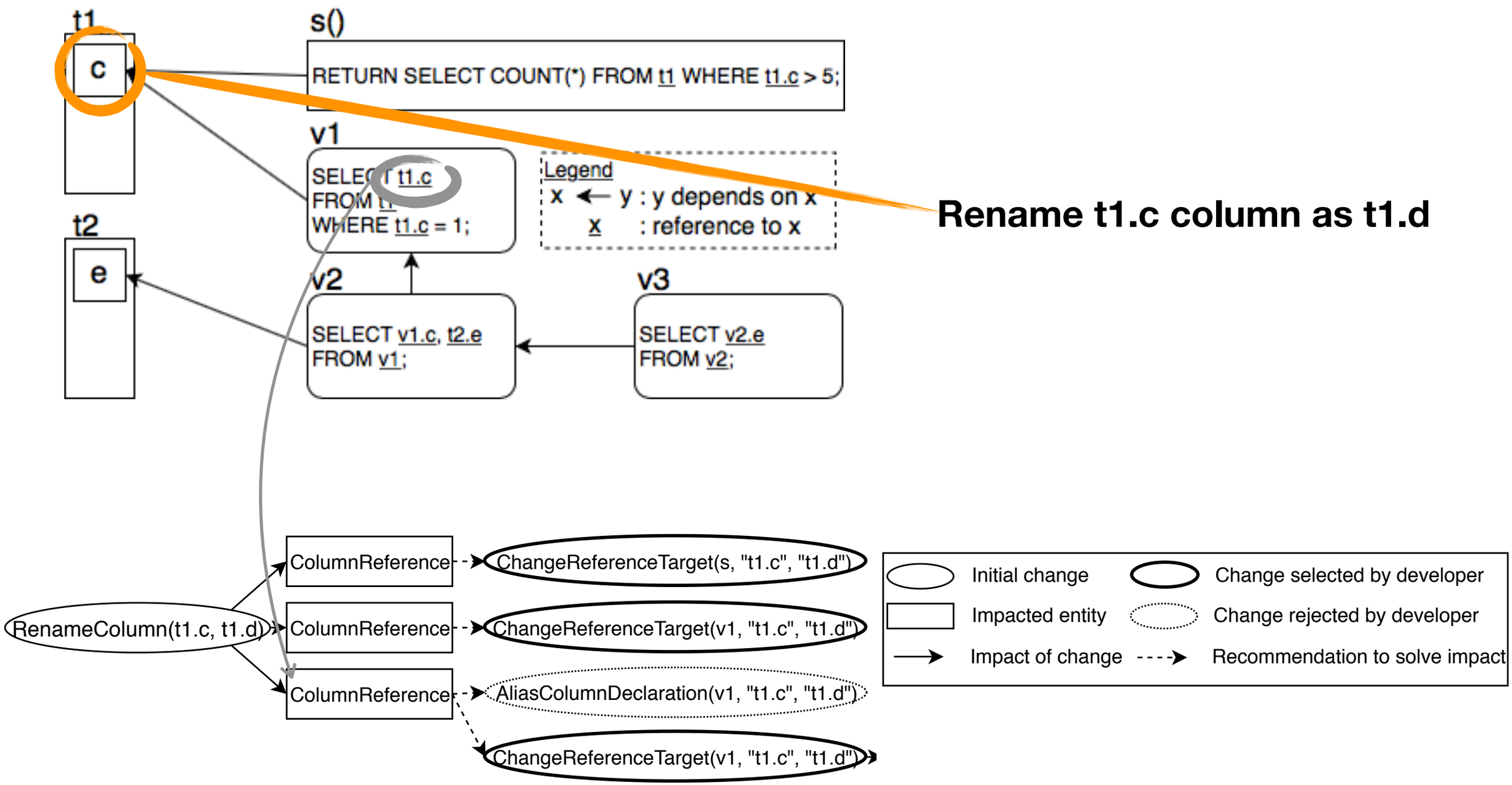
Impact computation & Recommendations selection



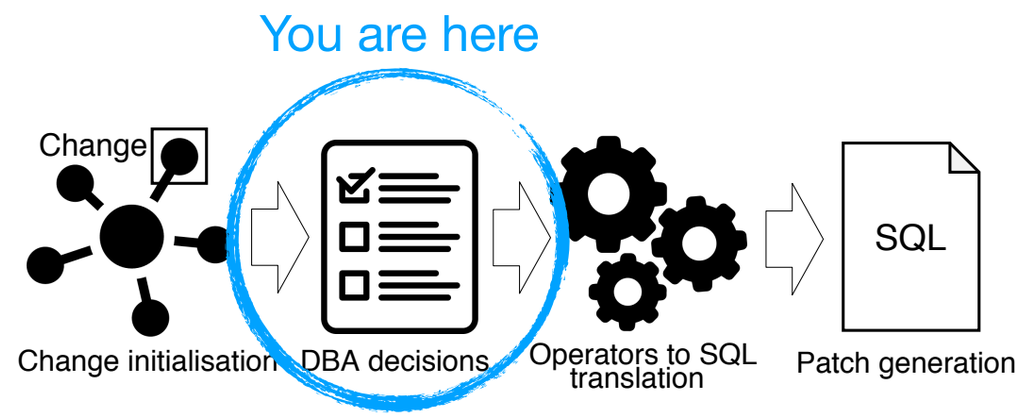


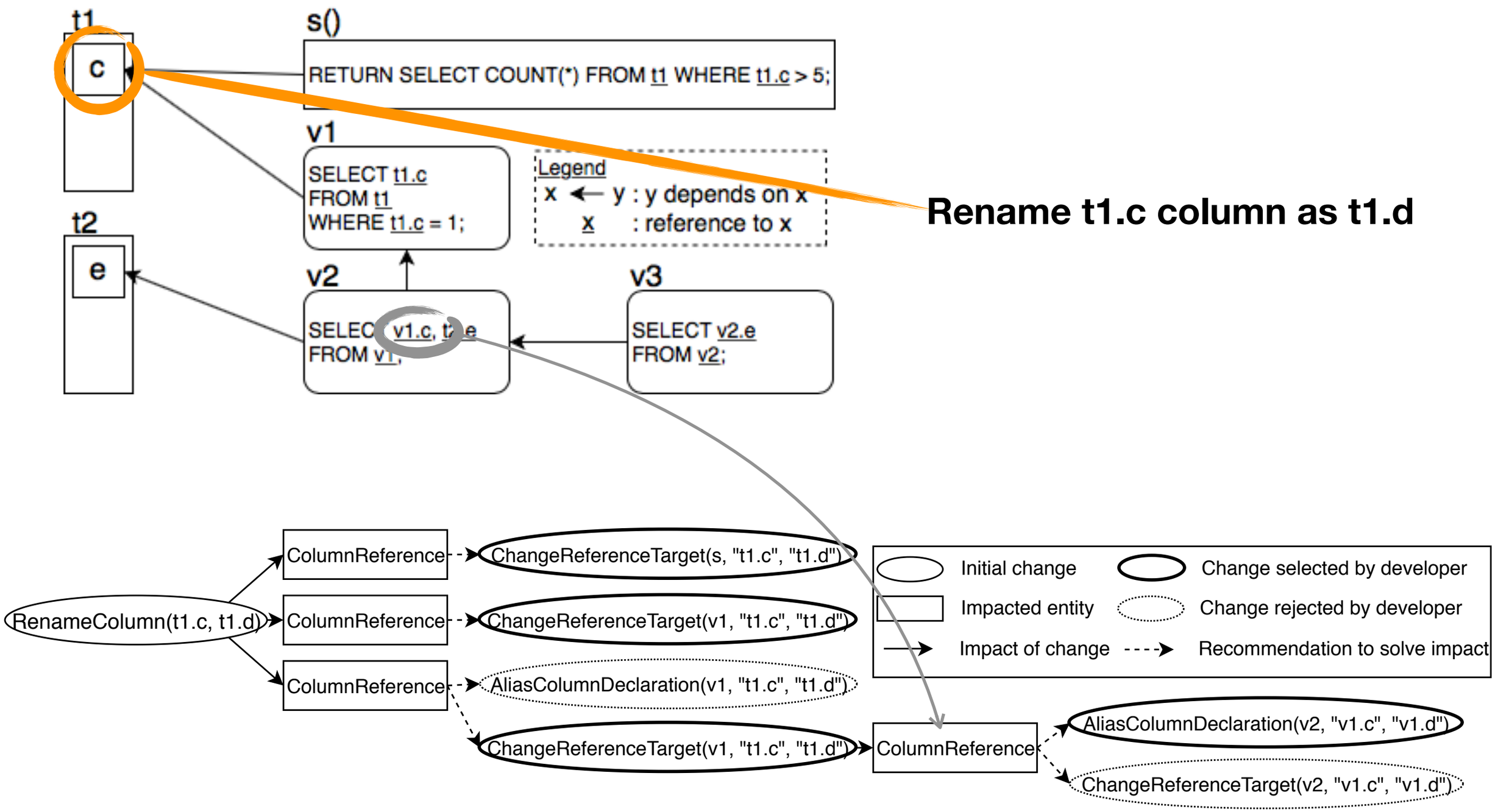
Impact computation & Recommendations selection



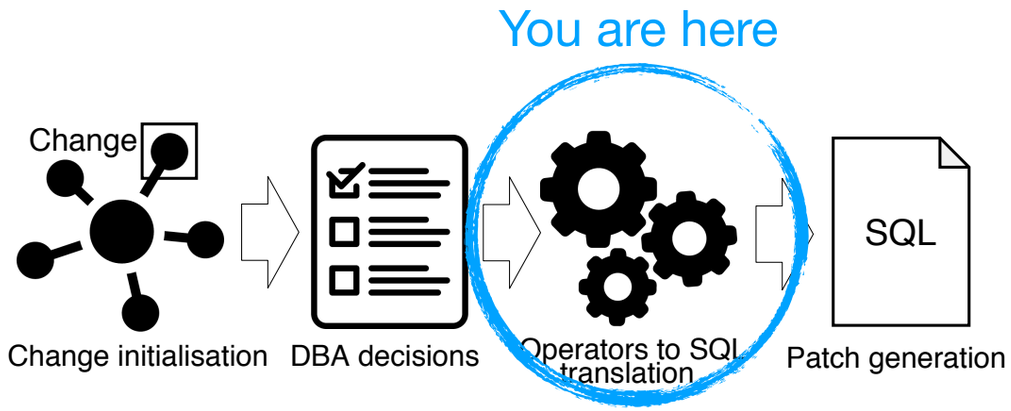


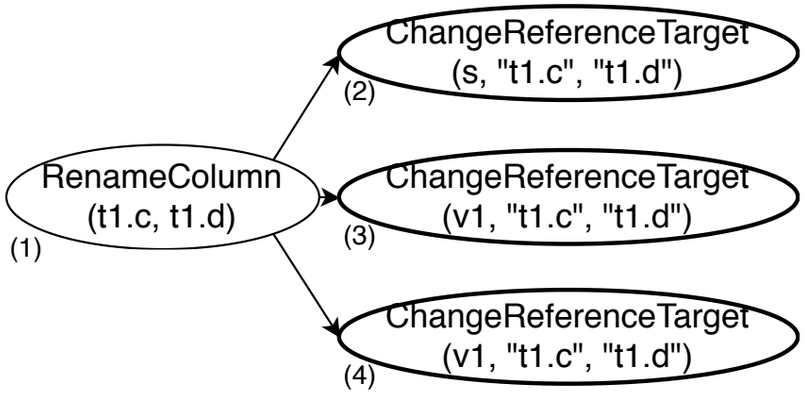
Impact computation & Recommendations selection



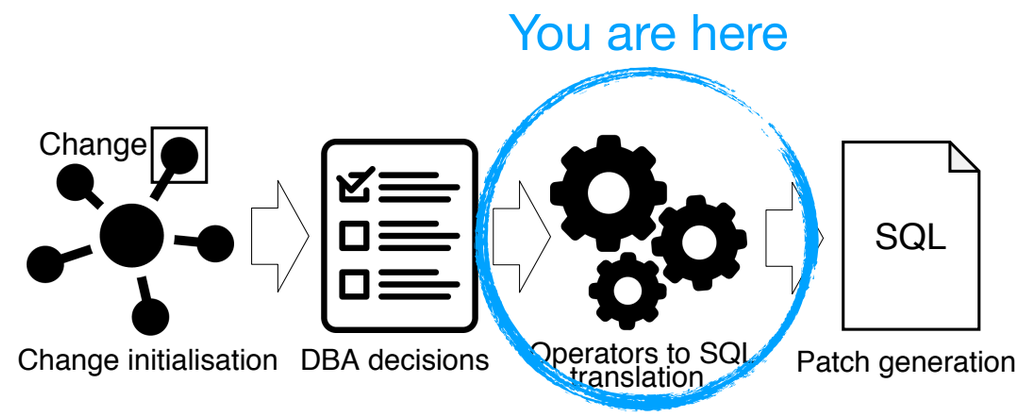


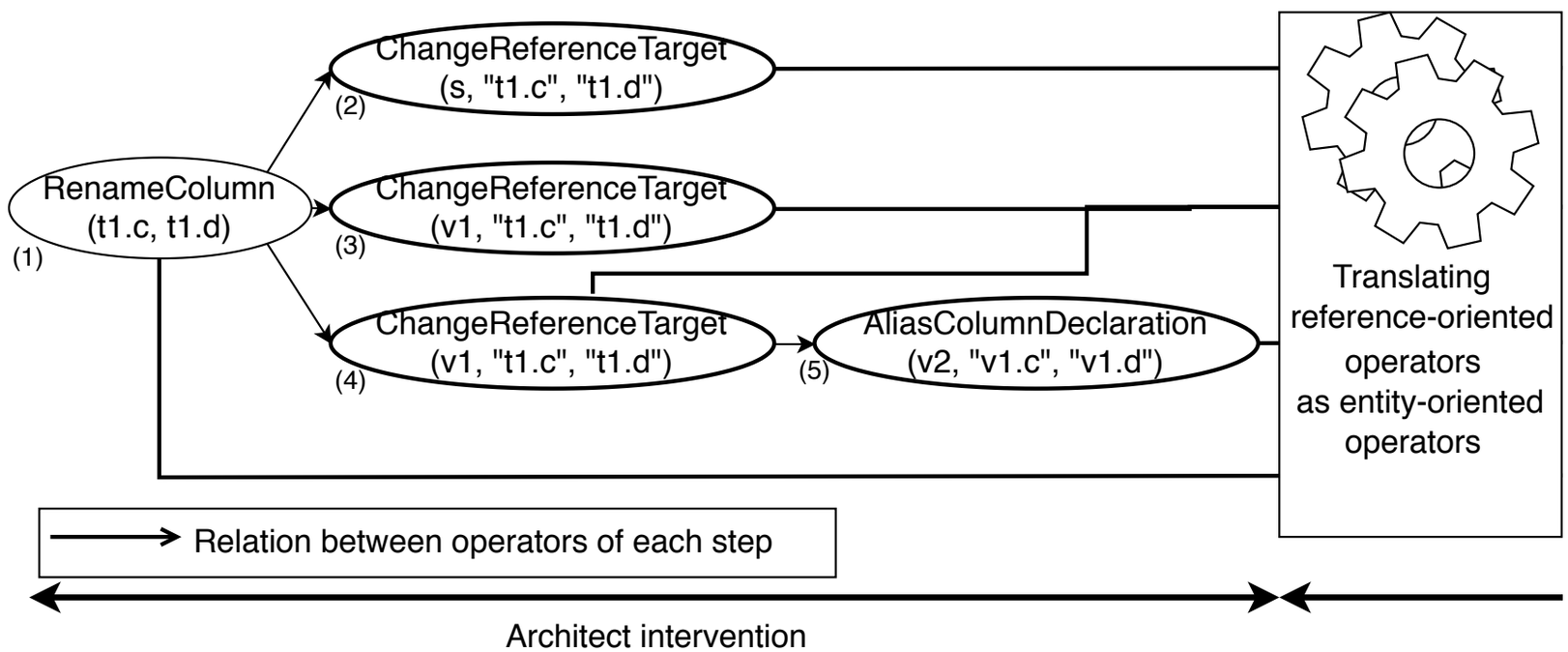
Architect choices compilation



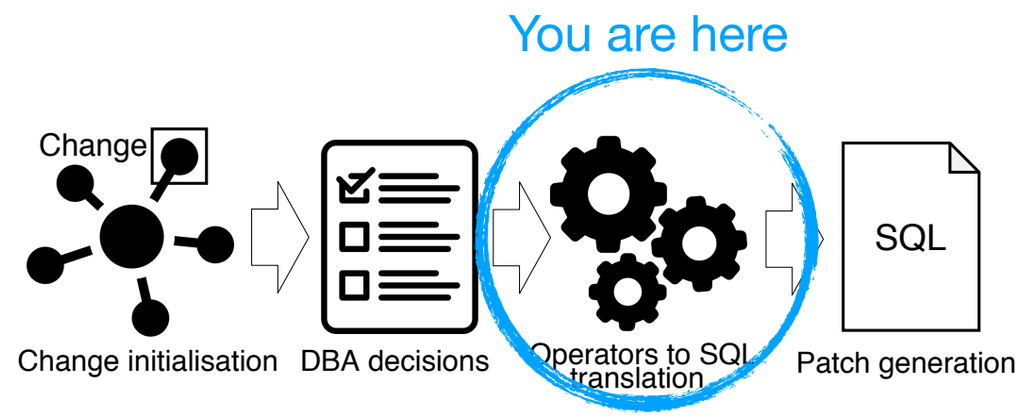


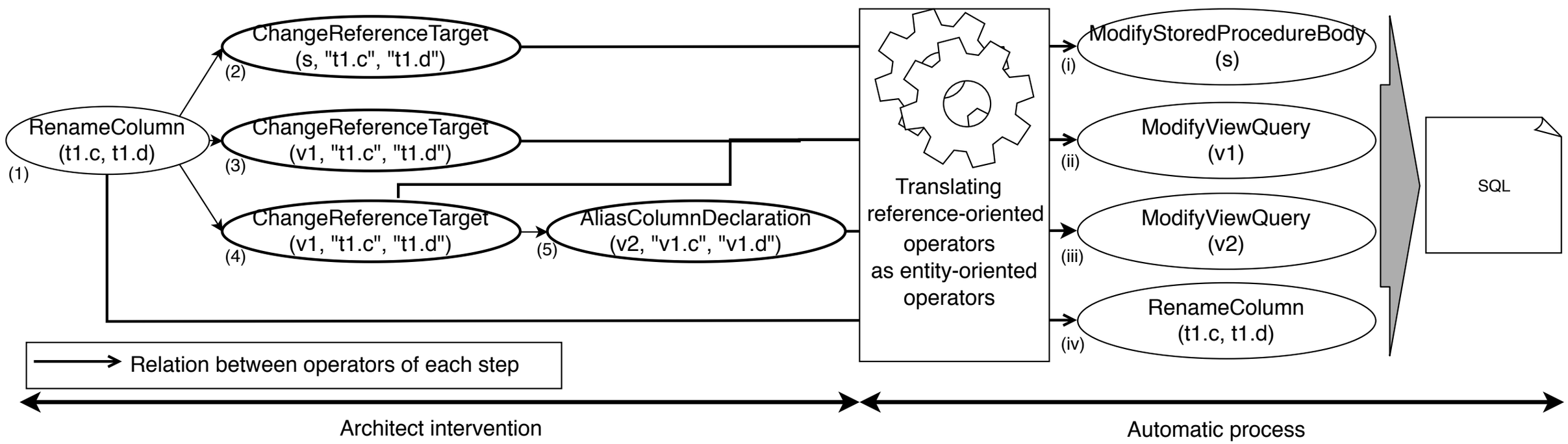
Architect choices compilation



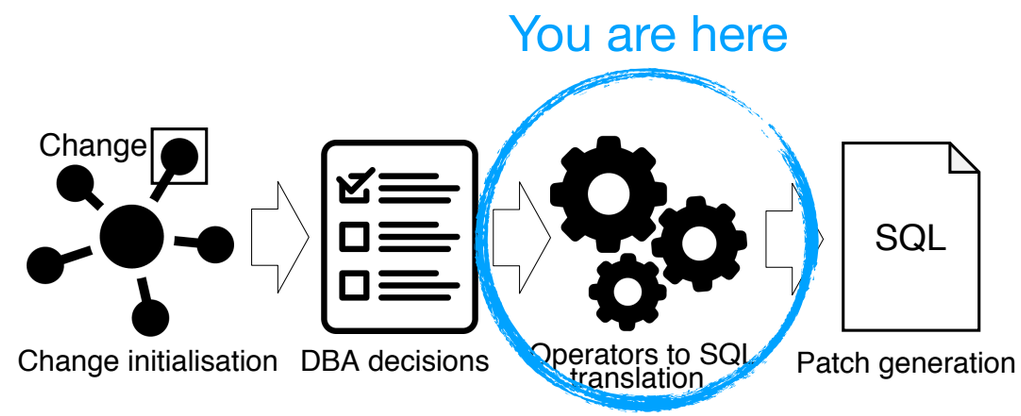


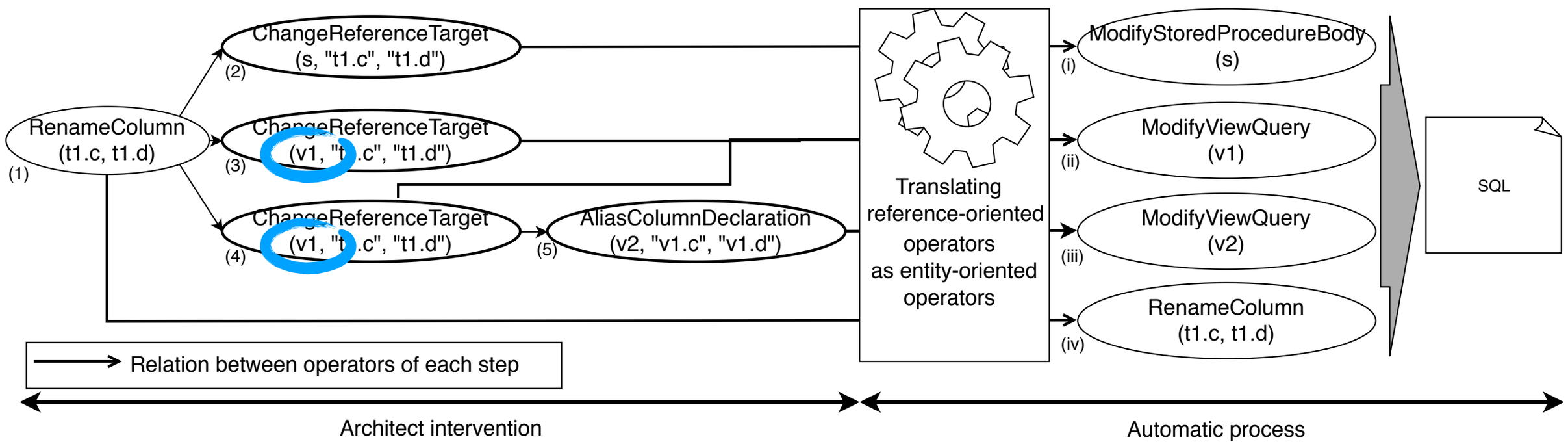
Architect choices compilation



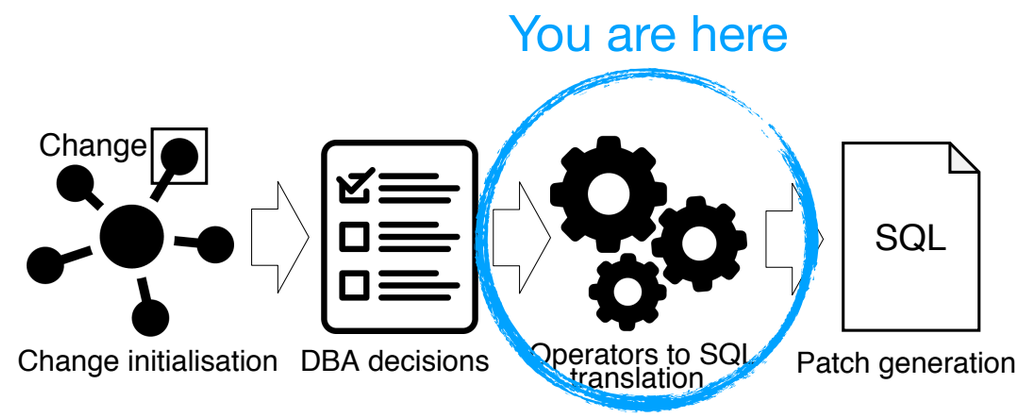


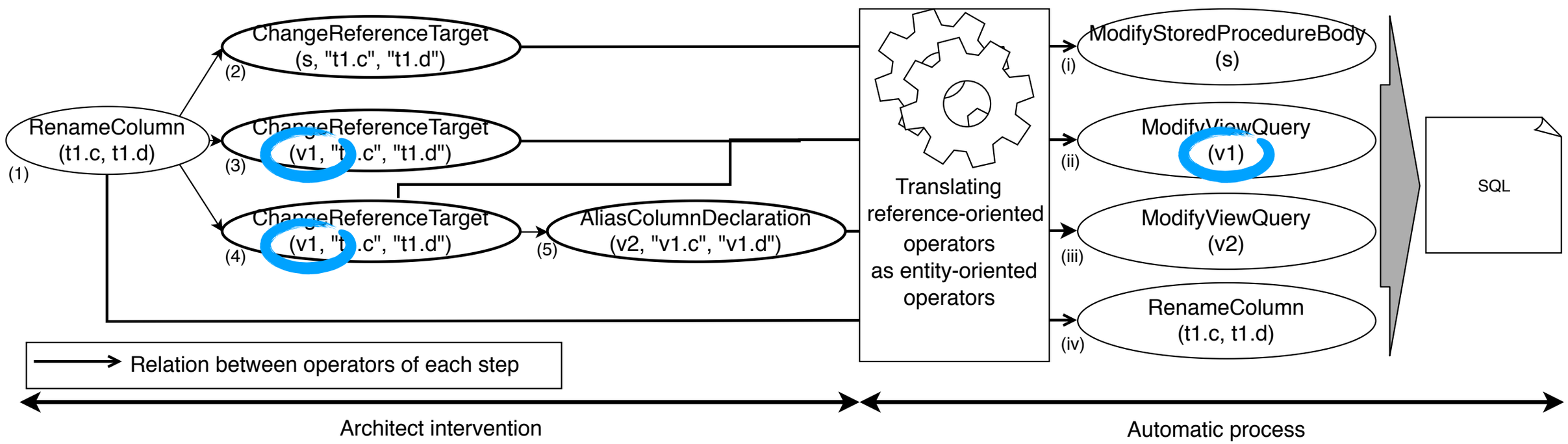
Architect choices compilation





Architect choices compilation





Experiment

- Information system to manage members of our university department
- 95 tables, 63 views, 109 stored procedures, 20 triggers
- Post-mortem analysis of a SQL patch on this database
- We observed trial-and-error process from DBA to find dependencies between entities in a previous study*
- 1h to achieve a script of 200 LOC / 19 statements

* Julien Delplanque, Anne Etien, Nicolas Anquetil, and Olivier Auverlot. Relational Database Schema Evolution: An Industrial Case Study. In 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2018.

Experimental protocol

1. **Extract initial** semantic **operators** from roadmap and SQL script comments

2. Take decisions using **DBA's policy** when multiple possibilities

3. **Execute** generated **SQL** script on a database in the same state as information system's **database before the migration**

Experiment - Initial semantic operators extracted

- RenameColumn(person.uid, login)
- RemoveFunction(key_for_uid(varchar))
- RemoveFunction(is_responsible_of(int4))
- RemoveFunction(is_responsible_of(int4,int4))
- RenameFunction(uid(integer), login(integer))
- RenameLocalVariable(login.uidperson, login.loginperson)
- RemoveView(test_member_view)

Results

- **15 decisions** taken concerning the choice between renaming and aliasing.
- **270 LOC** script with **27 SQL statements**.
- Generated script **executed without error**.
- **Single difference** between the dump of the original database and our clone: **a comment**.
- **Time to implement** the evolution using our tool: **15 min** (v.s. 60 min without tool).

Conclusion

- Approach developed to address 2 main constraints set by RDBMS:

  1. **No schema inconsistency is allowed during the evolution.**

  2. **Stored procedures bodies are not described by metadata.**

- Contributions:

  A. **Meta-model** for relational databases **easing impact computation**

  B. **Semi-automatic approach** to help in database **evolution**

  C. **Experiment** to **assess** the **effectiveness** of our approach

Future work

- Extend supported semantic operators
- Compare the performance of architects using or not our tool to achieve the same evolution
- Empirical study on multiple open-source projects using a relational database
  - Assess if our tool can reproduce various changes
  - Compare changes proposed by our tool with changes historically applied on the database