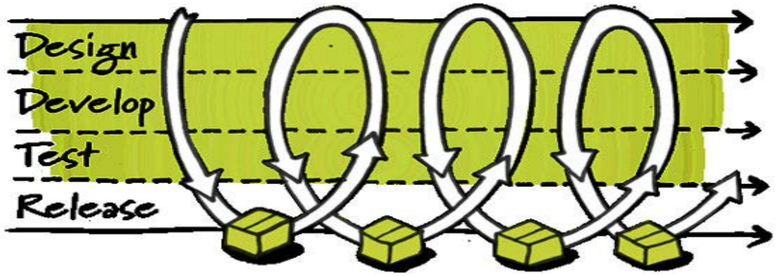
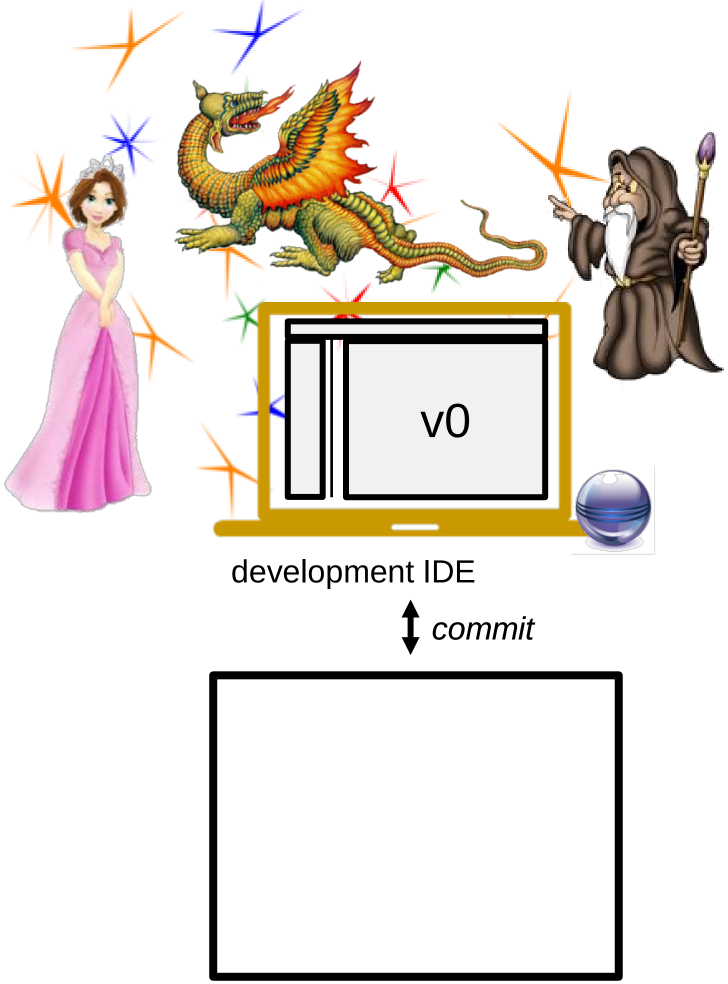
DATALUTION:
A TOOL FOR CONTINUOUS SCHEMA EVOLUTION IN NOSQL-BACKED WEB APPLICATIONS

STEFANIE SCHERZINGER
STEPHANIE SOMBACH
KATHARINA WIECH
MEIKE KLETTKE
UTA STÖRL

OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

datalution



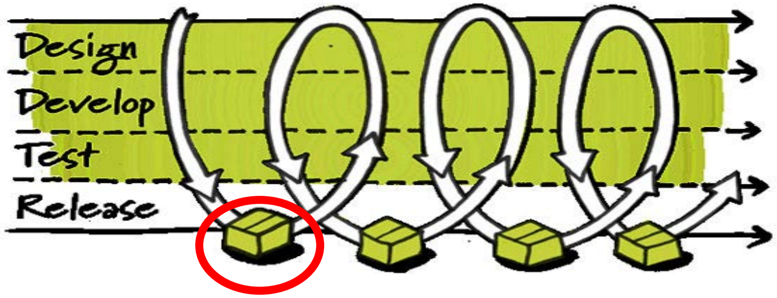


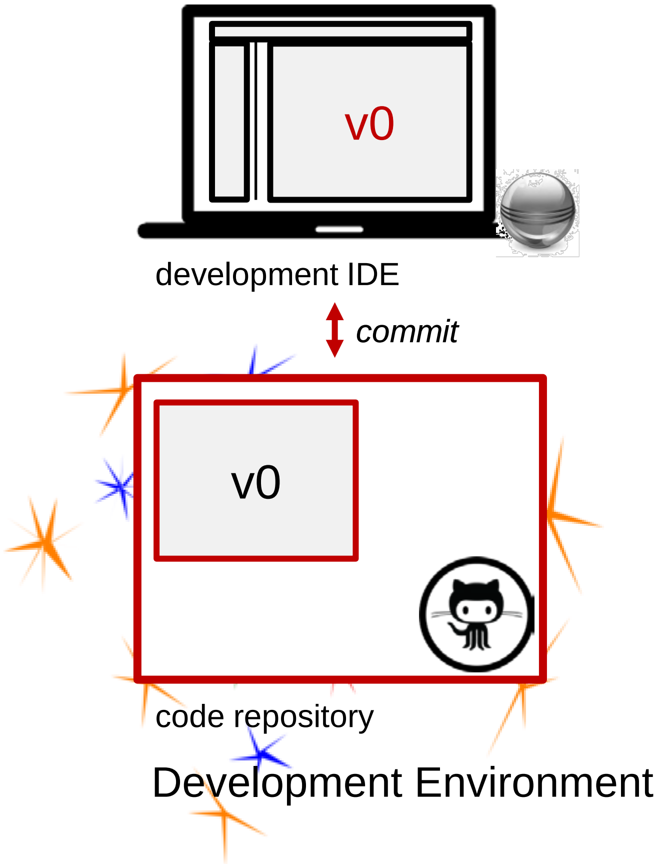
development IDE

commit

code repository

Development Environment




v0

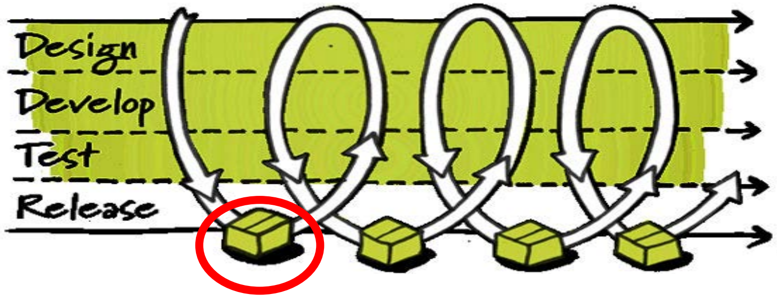
development IDE

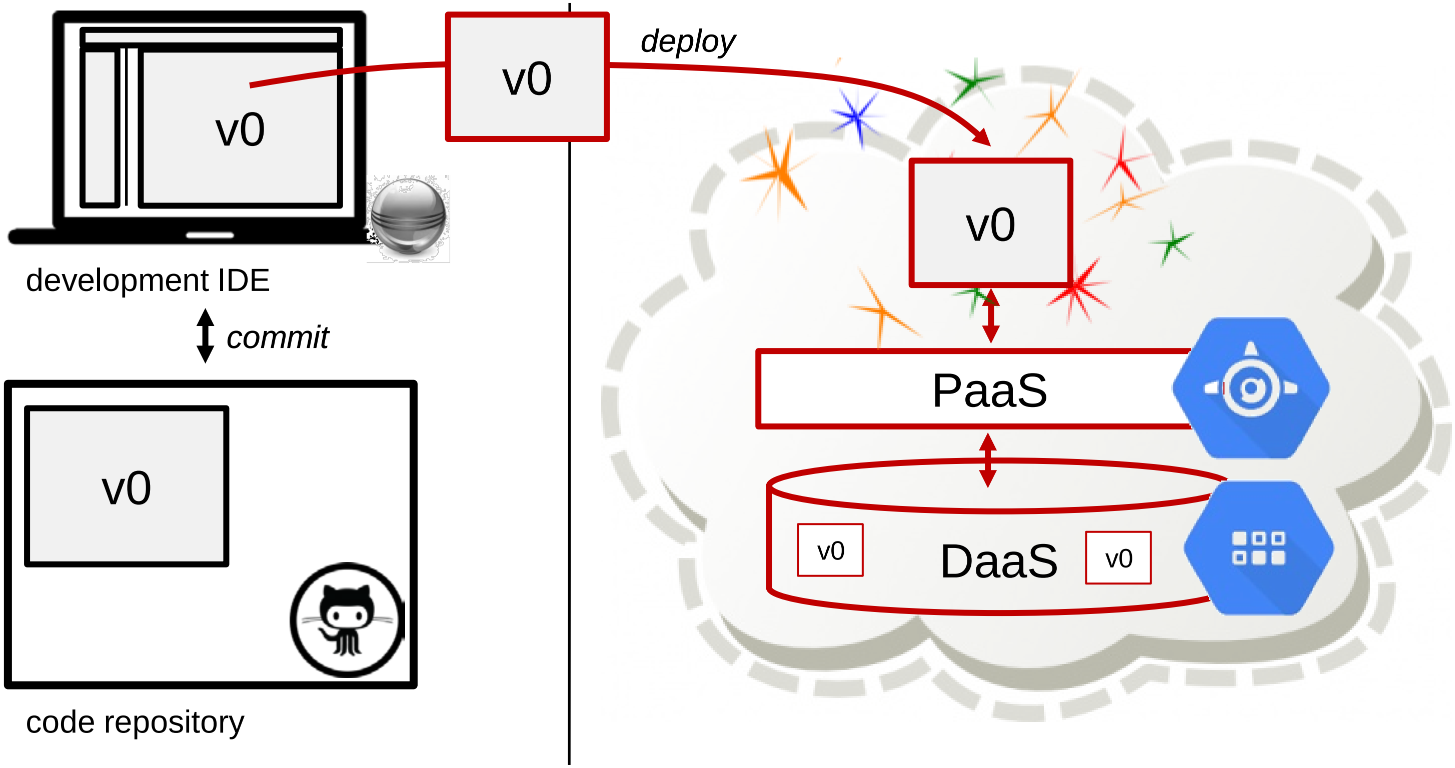
commit

v0

code repository

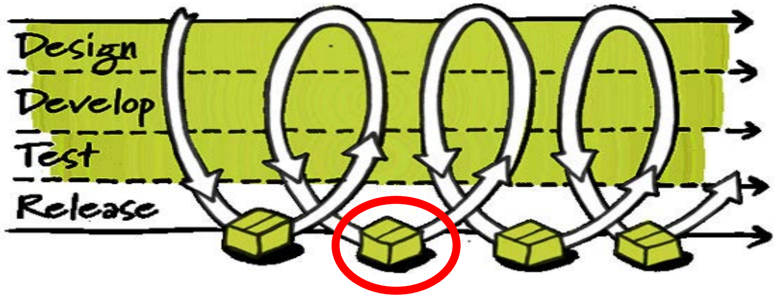
Development Environment

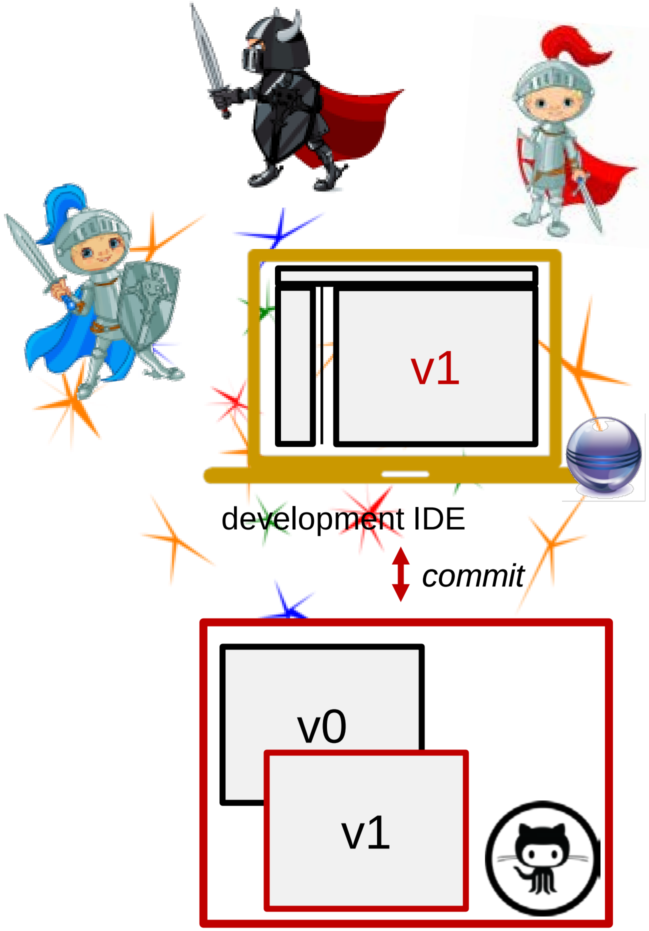




Development Environment

Production Environment

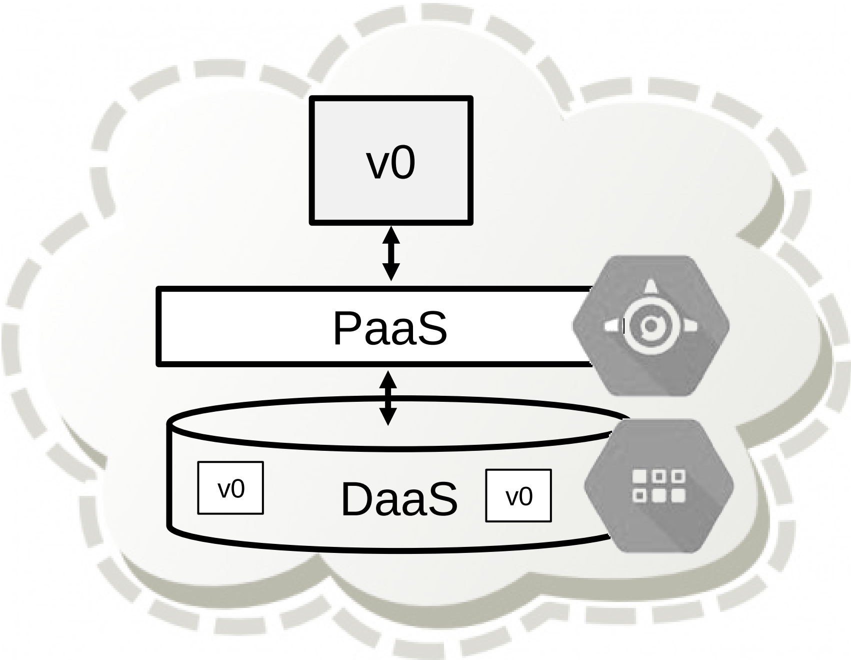




development IDE

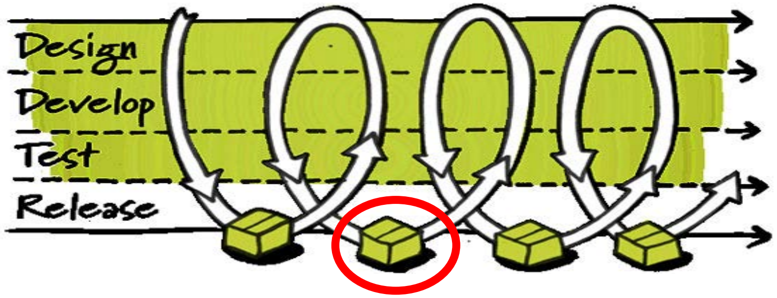
commit

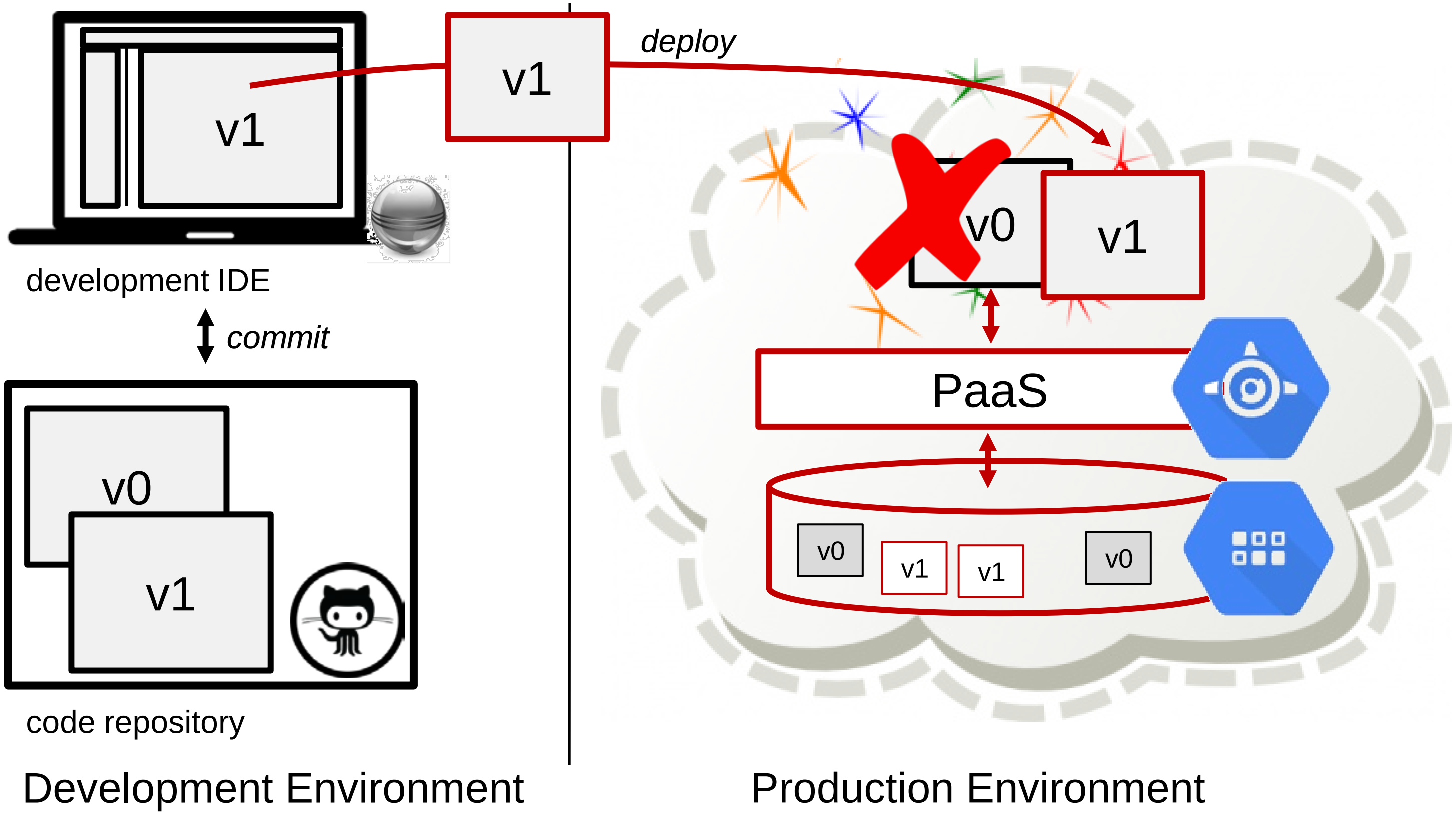
code repository



Development Environment

Production Environment




v1

development IDE

commit

v0

v1

code repository

v1

deploy

v0

v1

PaaS

v0

v1

v1

v0

Development Environment

Production Environment

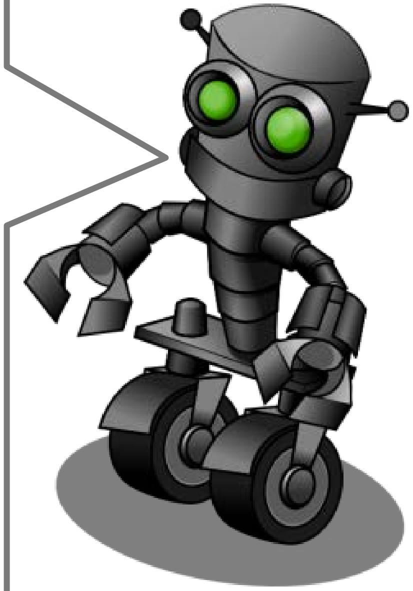
DESIDERATUM

The application code declares a schema.

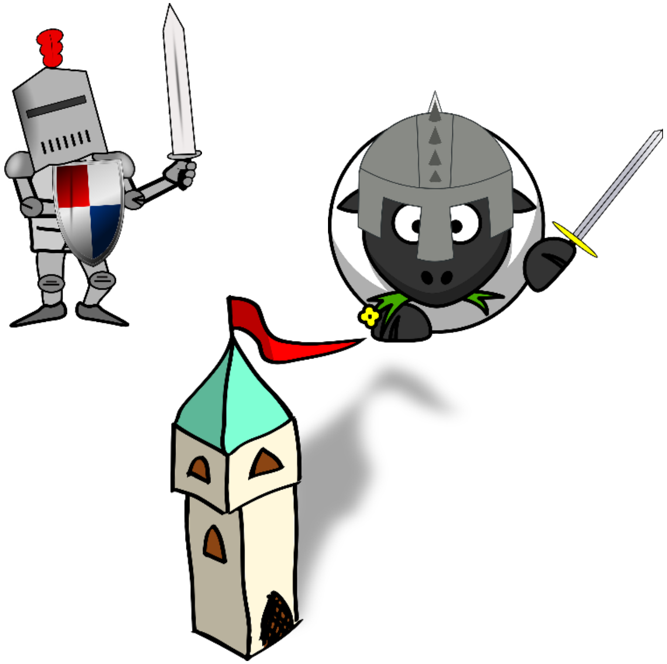
The application code evolves.

Thus, we need to address schema evolution:

- **Eager**
- **Lazy with Object-NoSQL Mappers**
- **Lazy with Datalution**



EXAMPLE: GAMING APPLICATION



Release 1

- `Player(ID, NAME)`
- `Mission(ID, TITLE, PID)`

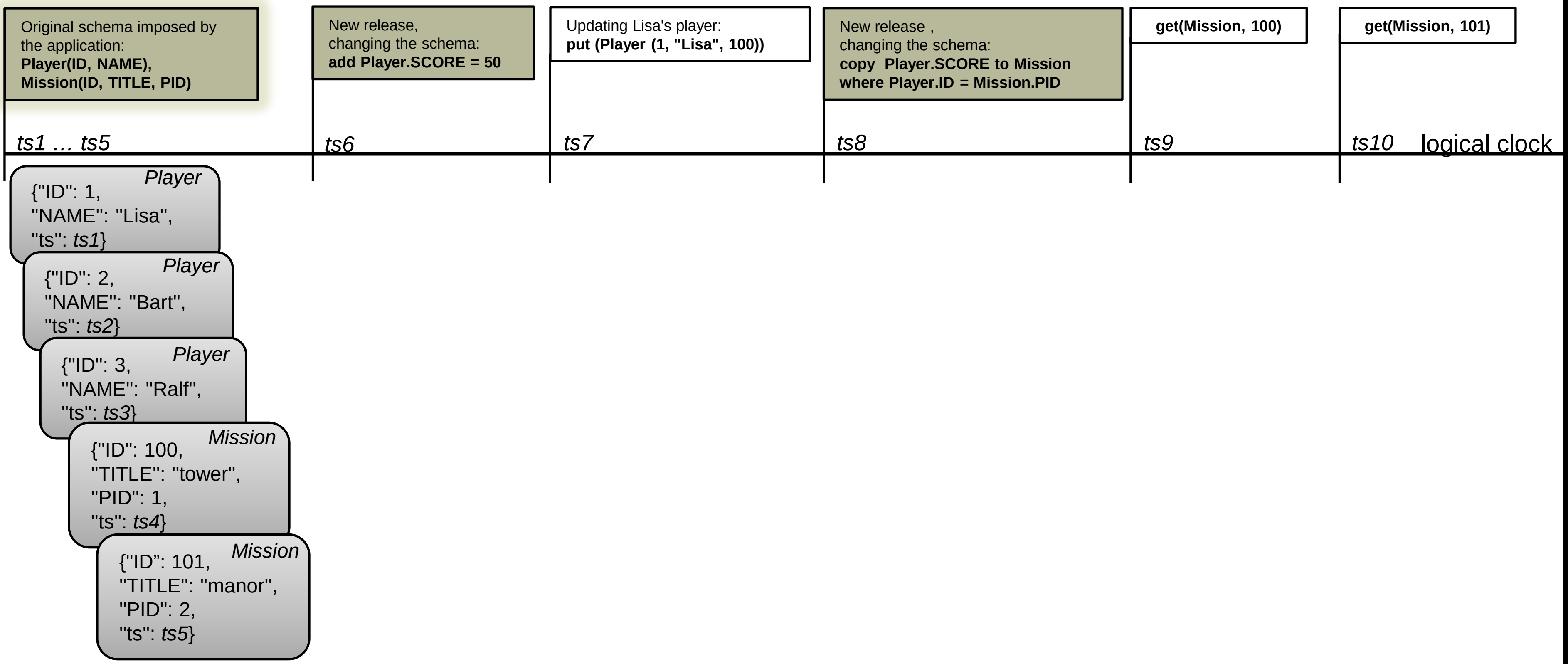
Release 2

- Players carry a property SCORE:
  `add Player.SCORE = 50`

Release 3

- Missions carry their player‘s score
  `copy Player.SCORE to Mission`
  `where Player.ID = Mission.PID`

EAGER MIGRATION



Original schema imposed by the application:
Player(ID, NAME),
Mission(ID, TITLE, PID)

New release,
changing the schema:
add Player.SCORE = 50

Updating Lisa's player:
put (Player (1, "Lisa", 100))

New release ,
changing the schema:
copy Player.SCORE to Mission
where Player.ID = Mission.PID

get(Mission, 100)

get(Mission, 101)

ts1 … ts5 *ts6* *ts7* *ts8* *ts9* *ts10* logical clock

Player
{"ID": 1,
"NAME": "Lisa",
"ts": *ts1*}

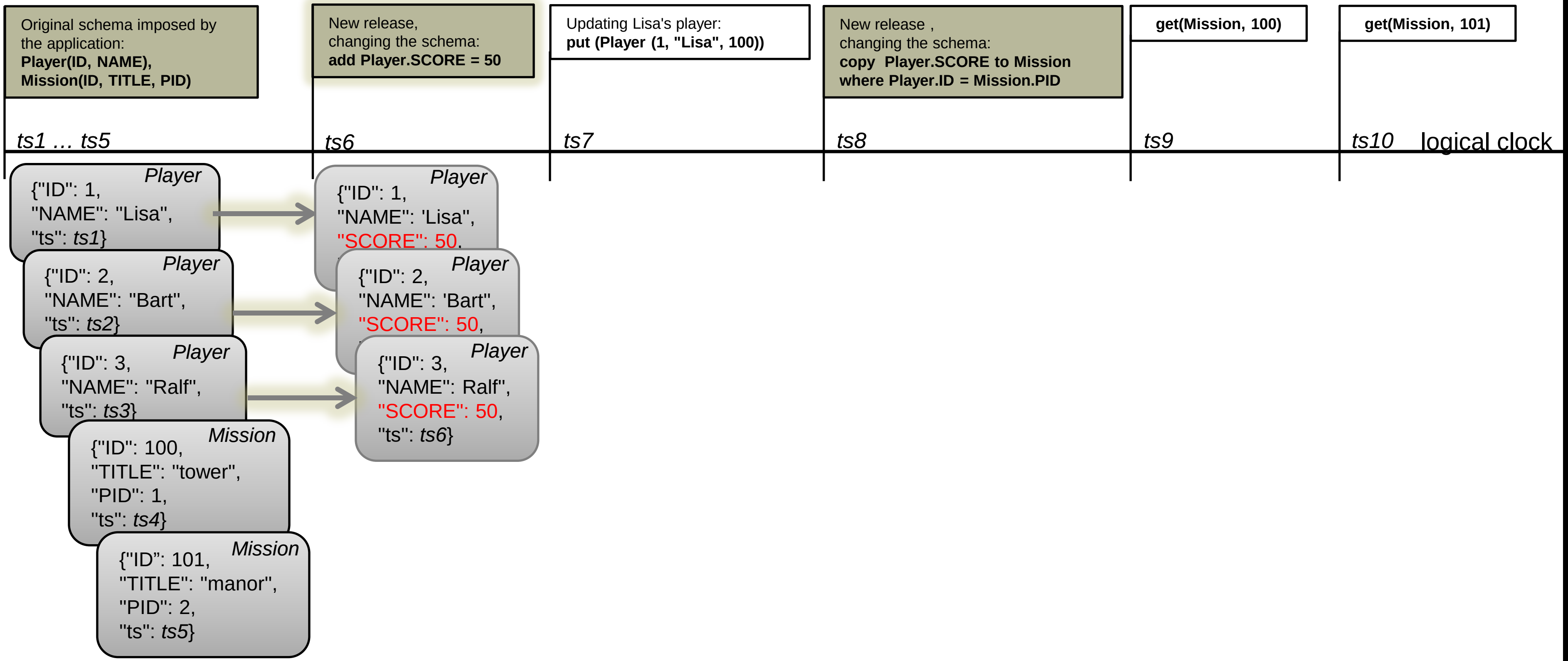
Player
{"ID": 2,
"NAME": "Bart",
"ts": *ts2*}

Player
{"ID": 3,
"NAME": "Ralf",
"ts": *ts3*}

Mission
{"ID": 100,
"TITLE": "tower",
"PID": 1,
"ts": *ts4*}

Mission
{"ID": 101,
"TITLE": "manor",
"PID": 2,
"ts": *ts5*}

EAGER MIGRATION



Original schema imposed by the application:
Player(ID, NAME),
Mission(ID, TITLE, PID)

New release, changing the schema:
add Player.SCORE = 50

Updating Lisa's player:
put (Player (1, "Lisa", 100))

New release , changing the schema:
copy Player.SCORE to Mission where Player.ID = Mission.PID

get(Mission, 100)

get(Mission, 101)

ts1 … *ts5*

ts6

ts7

ts8

ts9

ts10

logical clock

Player
{"ID": 1,
"NAME": "Lisa",
"ts": *ts1*}

Player
{"ID": 2,
"NAME": "Bart",
"ts": *ts2*}

Player
{"ID": 3,
"NAME": "Ralf",
"ts": *ts3*}

Mission
{"ID": 100,
"TITLE": "tower",
"PID": 1,
"ts": *ts4*}

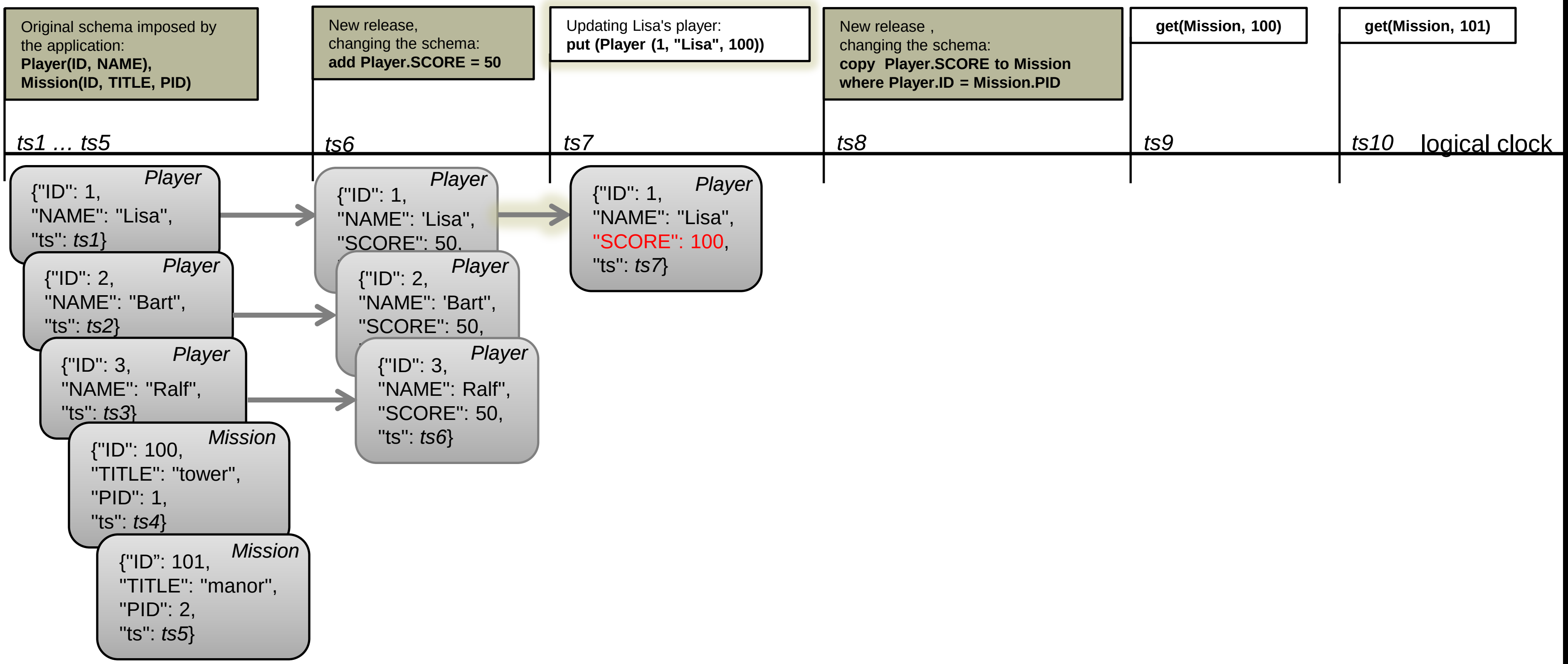
Mission
{"ID": 101,
"TITLE": "manor",
"PID": 2,
"ts": *ts5*}

Player
{"ID": 1,
"NAME": 'Lisa",
"SCORE": 50,

Player
{"ID": 2,
"NAME": 'Bart",
"SCORE": 50,

Player
{"ID": 3,
"NAME": Ralf",
"SCORE": 50,
"ts": *ts6*}

EAGER MIGRATION



Original schema imposed by the application:
Player(ID, NAME),
Mission(ID, TITLE, PID)

New release,
changing the schema:
add Player.SCORE = 50

Updating Lisa's player:
put (Player (1, "Lisa", 100))

New release ,
changing the schema:
copy Player.SCORE to Mission
where Player.ID = Mission.PID

get(Mission, 100)

get(Mission, 101)

ts1 … *ts5*

ts6

ts7

ts8

ts9

ts10

logical clock

Player
{"ID": 1,
"NAME": "Lisa",
"ts": *ts1*}

Player
{"ID": 2,
"NAME": "Bart",
"ts": *ts2*}

Player
{"ID": 3,
"NAME": "Ralf",
"ts": *ts3*}

Mission
{"ID": 100,
"TITLE": "tower",
"PID": 1,
"ts": *ts4*}

Mission
{"ID": 101,
"TITLE": "manor",
"PID": 2,
"ts": *ts5*}

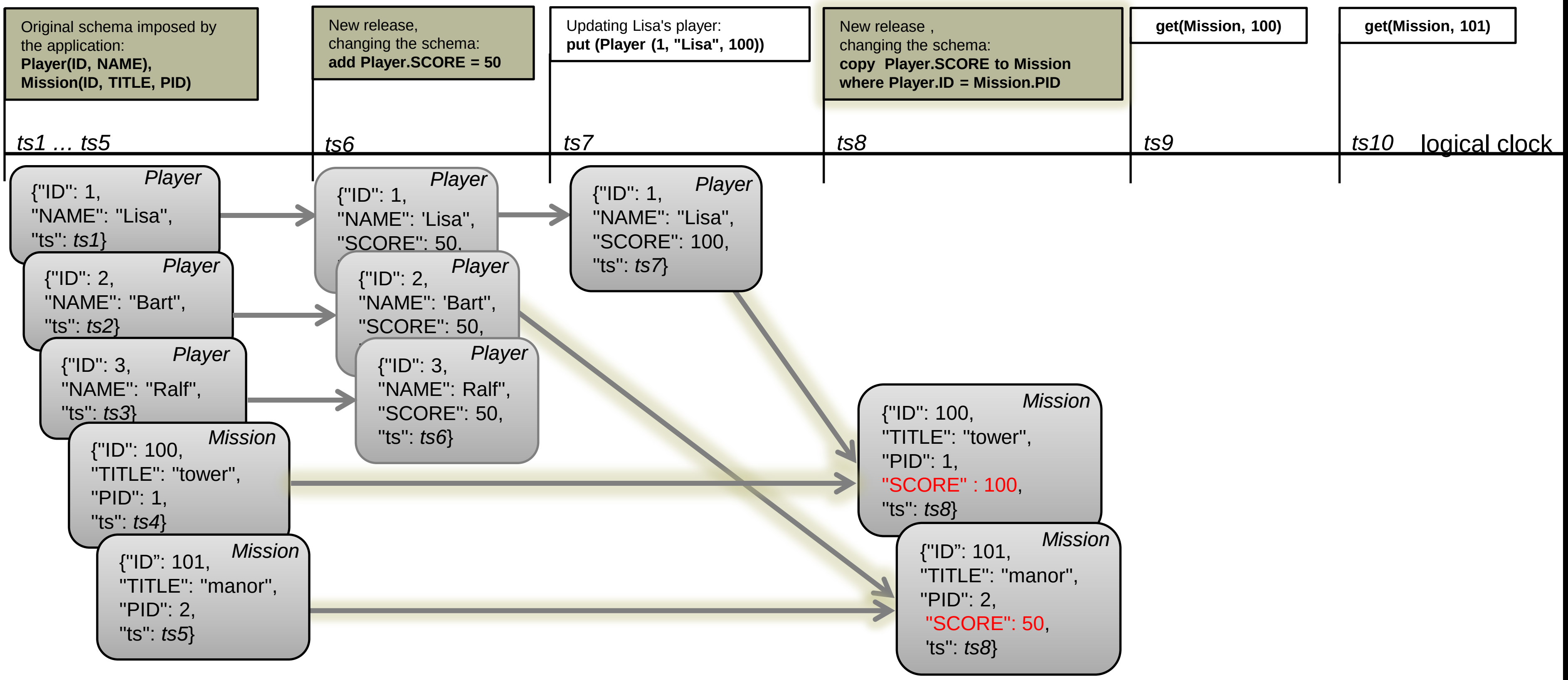
Player
{"ID": 1,
"NAME": 'Lisa",
"SCORE": 50,

Player
{"ID": 2,
"NAME": 'Bart",
"SCORE": 50,

Player
{"ID": 3,
"NAME": Ralf",
"SCORE": 50,
"ts": *ts6*}

Player
{"ID": 1,
"NAME": "Lisa",
"SCORE": 100,
"ts": *ts7*}

EAGER MIGRATION



Original schema imposed by the application:
Player(ID, NAME),
Mission(ID, TITLE, PID)

New release,
changing the schema:
add Player.SCORE = 50

Updating Lisa's player:
put (Player (1, "Lisa", 100))

New release ,
changing the schema:
copy Player.SCORE to Mission
where Player.ID = Mission.PID

get(Mission, 100)

get(Mission, 101)

ts1 … ts5 *ts6* *ts7* *ts8* *ts9* *ts10* logical clock

Player
{"ID": 1,
"NAME": "Lisa",
"ts": *ts1*}

Player
{"ID": 2,
"NAME": "Bart",
"ts": *ts2*}

Player
{"ID": 3,
"NAME": "Ralf",
"ts": *ts3*}

Mission
{"ID": 100,
"TITLE": "tower",
"PID": 1,
"ts": *ts4*}

Mission
{"ID”: 101,
"TITLE": "manor",
"PID": 2,
"ts": *ts5*}

Player
{"ID": 1,
"NAME": 'Lisa",
"SCORE": 50,

Player
{"ID": 2,
"NAME": 'Bart",
"SCORE": 50,

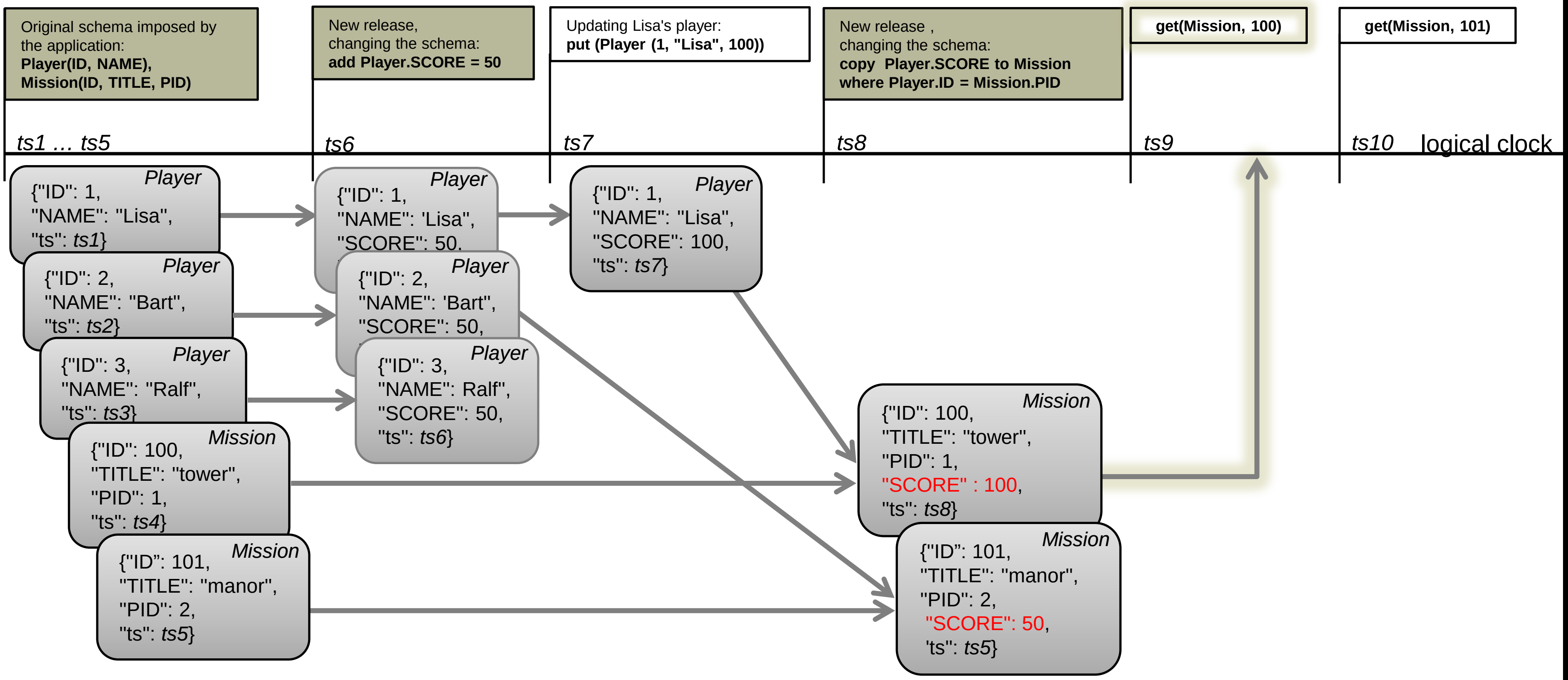
Player
{"ID": 3,
"NAME": Ralf",
"SCORE": 50,
"ts": *ts6*}

Player
{"ID": 1,
"NAME": "Lisa",
"SCORE": 100,
"ts": *ts7*}

Mission
{"ID": 100,
"TITLE": "tower",
"PID": 1,
"SCORE" : 100,
"ts": *ts8*}

Mission
{"ID”: 101,
"TITLE": "manor",
"PID": 2,
"SCORE": 50,
'ts": *ts8*}

EAGER MIGRATION



Original schema imposed by the application:
Player(ID, NAME),
Mission(ID, TITLE, PID)

New release,
changing the schema:
add Player.SCORE = 50

Updating Lisa's player:
put (Player (1, "Lisa", 100))

New release ,
changing the schema:
copy Player.SCORE to Mission
where Player.ID = Mission.PID

get(Mission, 100)

get(Mission, 101)

ts1 … ts5

ts6

ts7

ts8

ts9

ts10

logical clock

Player
{"ID": 1,
"NAME": "Lisa",
"ts": *ts1*}

Player
{"ID": 2,
"NAME": "Bart",
"ts": *ts2*}

Player
{"ID": 3,
"NAME": "Ralf",
"ts": *ts3*}

Mission
{"ID": 100,
"TITLE": "tower",
"PID": 1,
"ts": *ts4*}

Mission
{"ID": 101,
"TITLE": "manor",
"PID": 2,
"ts": *ts5*}

Player
{"ID": 1,
"NAME": 'Lisa",
"SCORE": 50,

Player
{"ID": 2,
"NAME": 'Bart",
"SCORE": 50,

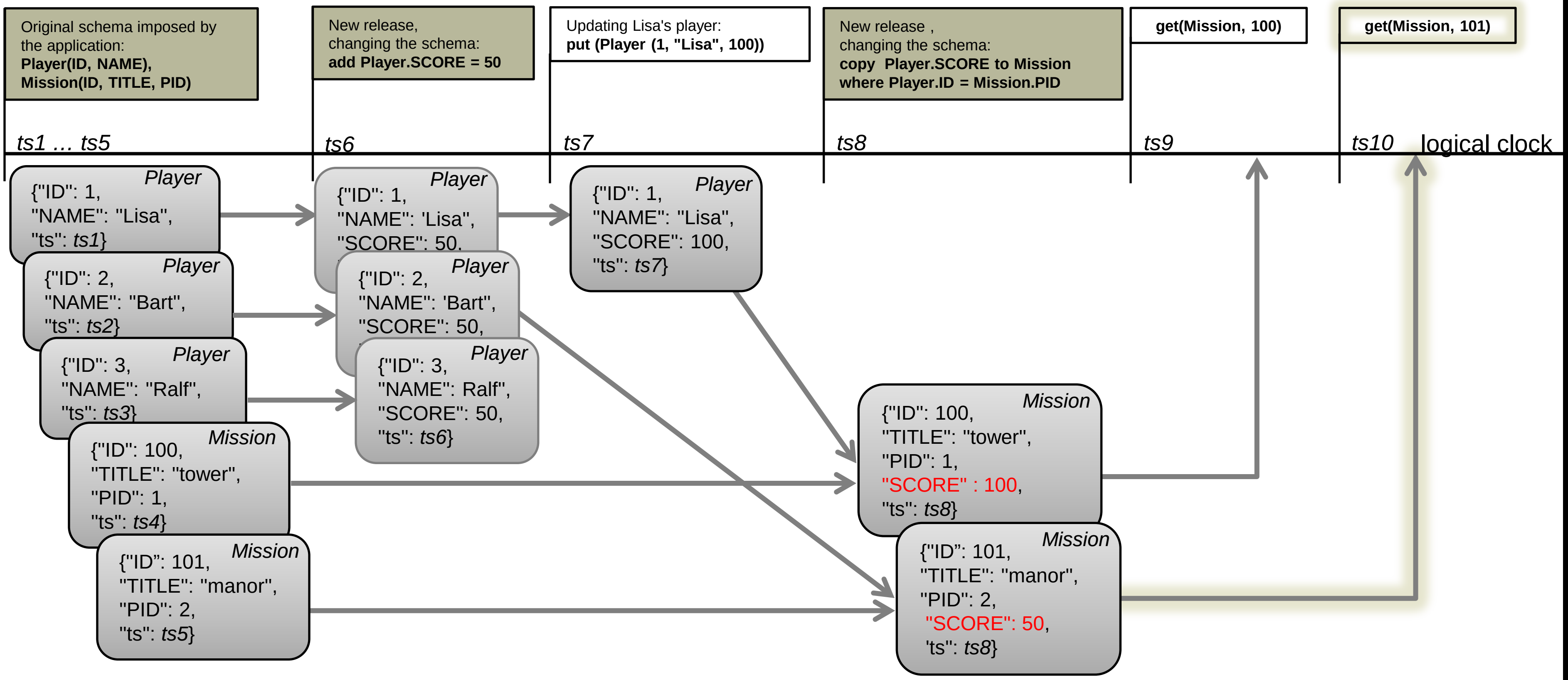
Player
{"ID": 3,
"NAME": Ralf",
"SCORE": 50,
"ts": *ts6*}

Player
{"ID": 1,
"NAME": "Lisa",
"SCORE": 100,
"ts": *ts7*}

Mission
{"ID": 100,
"TITLE": "tower",
"PID": 1,
"SCORE" : 100,
"ts": *ts8*}

Mission
{"ID": 101,
"TITLE": "manor",
"PID": 2,
"SCORE": 50,
'ts": *ts5*}

EAGER MIGRATION



Original schema imposed by the application:
Player(ID, NAME),
Mission(ID, TITLE, PID)

New release, changing the schema:
add Player.SCORE = 50

Updating Lisa's player:
put (Player (1, "Lisa", 100))

New release , changing the schema:
copy Player.SCORE to Mission
where Player.ID = Mission.PID

get(Mission, 100)

get(Mission, 101)

Player
{"ID": 1,
"NAME": "Lisa",
"ts": *ts1*}

Player
{"ID": 2,
"NAME": "Bart",
"ts": *ts2*}

Player
{"ID": 3,
"NAME": "Ralf",
"ts": *ts3*}

Mission
{"ID": 100,
"TITLE": "tower",
"PID": 1,
"ts": *ts4*}

Mission
{"ID": 101,
"TITLE": "manor",
"PID": 2,
"ts": *ts5*}

Player
{"ID": 1,
"NAME": 'Lisa",
"SCORE": 50,

Player
{"ID": 2,
"NAME": 'Bart",
"SCORE": 50,

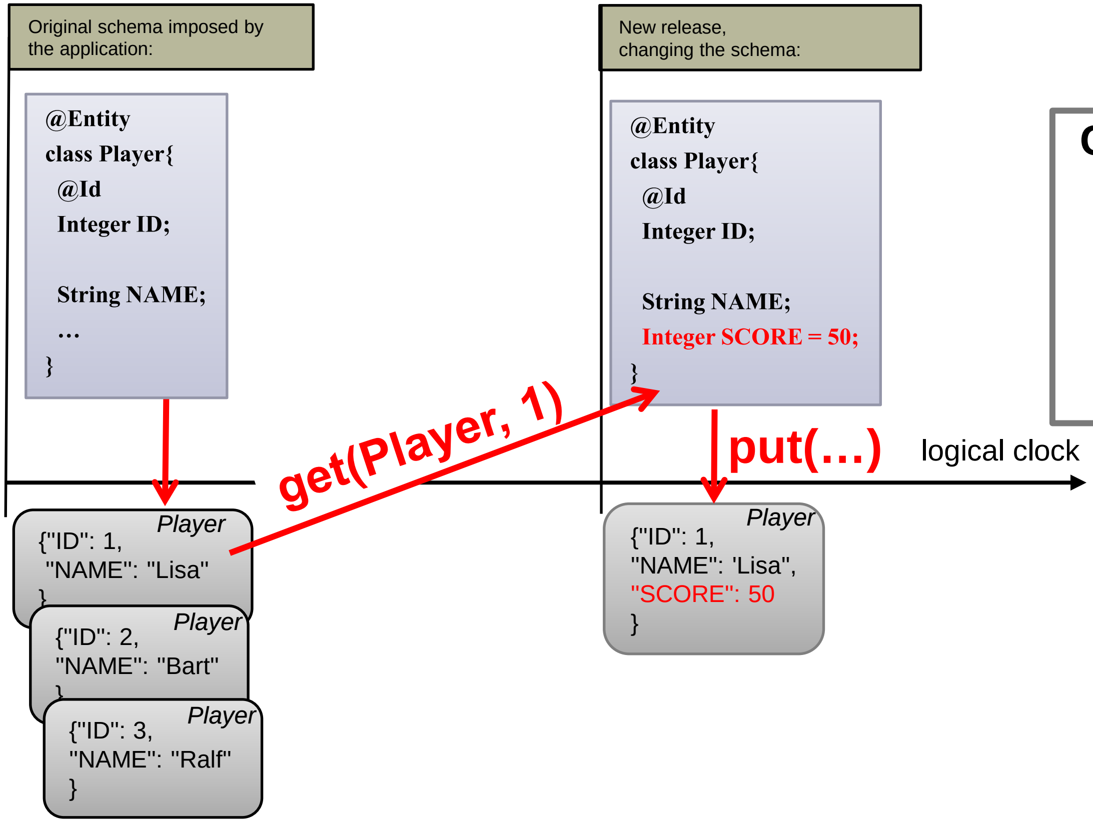
Player
{"ID": 3,
"NAME": Ralf",
"SCORE": 50,
"ts": *ts6*}

Player
{"ID": 1,
"NAME": "Lisa",
"SCORE": 100,
"ts": *ts7*}

Mission
{"ID": 100,
"TITLE": "tower",
"PID": 1,
"SCORE" : 100,
"ts": *ts8*}

Mission
{"ID": 101,
"TITLE": "manor",
"PID": 2,
"SCORE": 50,
'ts": *ts8*}

LAZY EVOLUTION
WITH OBJECT-NOSQL MAPPERS



Original schema imposed by the application:

```
@Entity
class Player{
  @Id
  Integer ID;

  String NAME;
  …
}
```

```
Player
{"ID": 1,
 "NAME": "Lisa"
}
```

```
Player
{"ID": 2,
"NAME": "Bart"
}
```

```
Player
{"ID": 3,
"NAME": "Ralf"
}
```

New release, changing the schema:

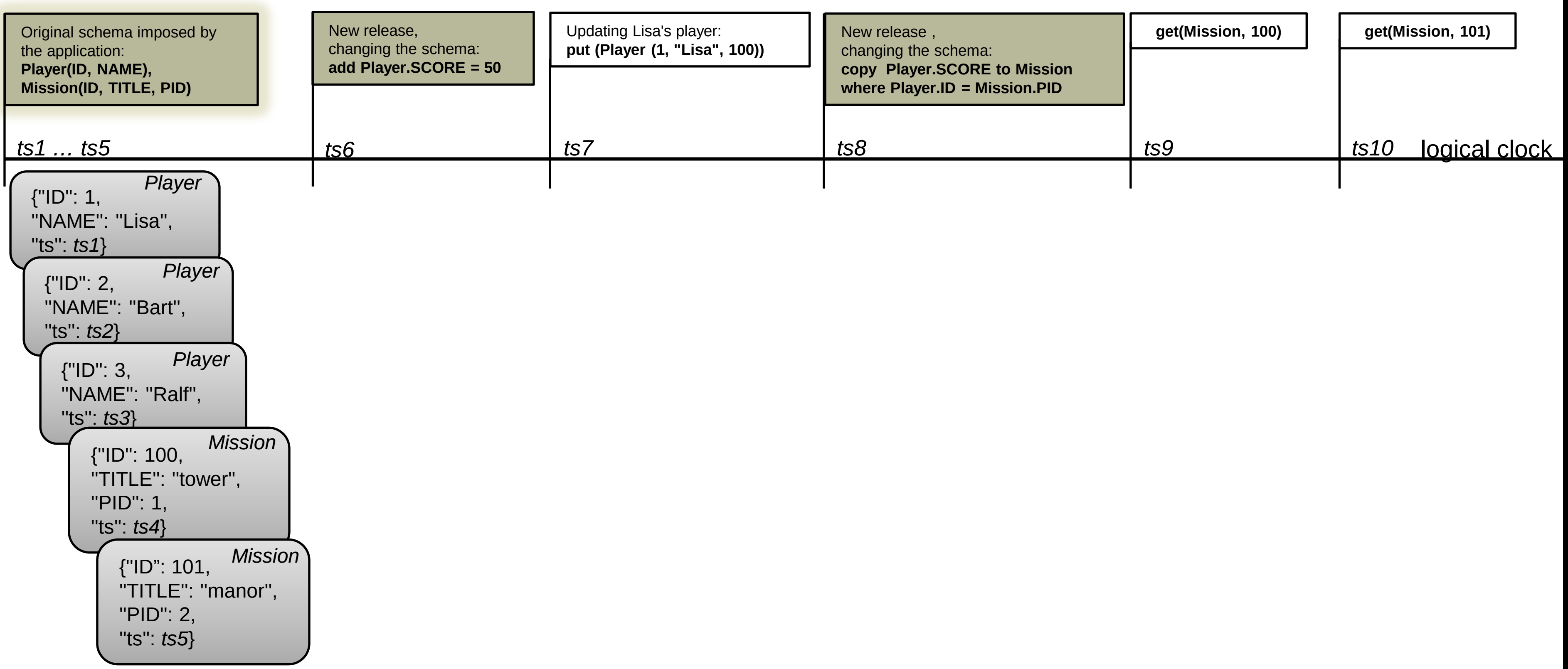
```
@Entity
class Player{
  @Id
  Integer ID;

  String NAME;
  Integer SCORE = 50;
}
```

```
Player
{"ID": 1,
"NAME": 'Lisa",
"SCORE": 50
}
```

Convenient "quick fix" for simple changes.

Long-term: Maintenance nightmare.

LAZY MIGRATION IN DATALUTION



Original schema imposed by the application:
Player(ID, NAME),
Mission(ID, TITLE, PID)

New release,
changing the schema:
add Player.SCORE = 50

Updating Lisa's player:
put (Player (1, "Lisa", 100))

New release ,
changing the schema:
copy Player.SCORE to Mission
where Player.ID = Mission.PID

get(Mission, 100)

get(Mission, 101)

ts1 … ts5

ts6

ts7

ts8

ts9

ts10

logical clock

Player
{"ID": 1,
"NAME": "Lisa",
"ts": *ts1*}

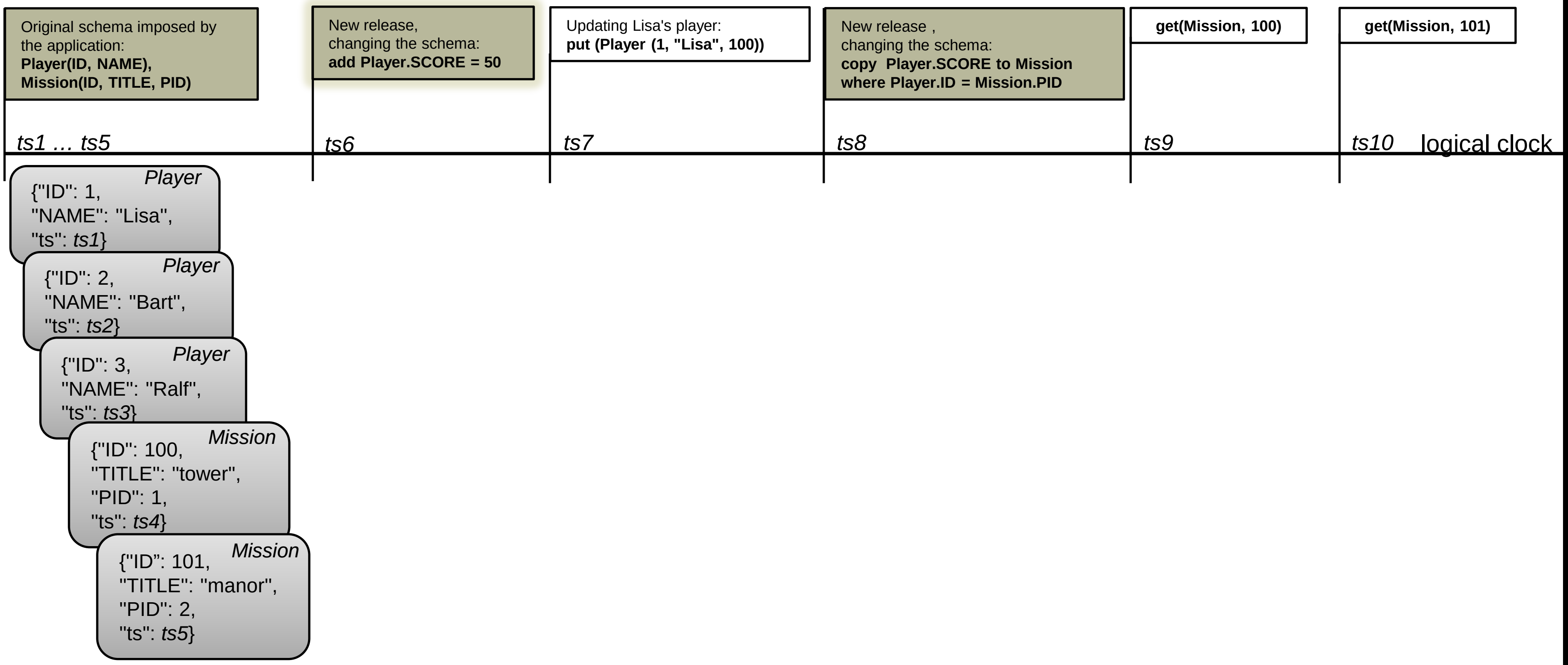
Player
{"ID": 2,
"NAME": "Bart",
"ts": *ts2*}

Player
{"ID": 3,
"NAME": "Ralf",
"ts": *ts3*}

Mission
{"ID": 100,
"TITLE": "tower",
"PID": 1,
"ts": *ts4*}

Mission
{"ID": 101,
"TITLE": "manor",
"PID": 2,
"ts": *ts5*}

LAZY MIGRATION IN DATALUTION



Original schema imposed by the application:
Player(ID, NAME),
Mission(ID, TITLE, PID)

New release,
changing the schema:
add Player.SCORE = 50

Updating Lisa's player:
put (Player (1, "Lisa", 100))

New release ,
changing the schema:
copy Player.SCORE to Mission
where Player.ID = Mission.PID

get(Mission, 100)

get(Mission, 101)

ts1 … ts5 *ts6* *ts7* *ts8* *ts9* *ts10* logical clock

Player
{"ID": 1,
"NAME": "Lisa",
"ts": *ts1*}

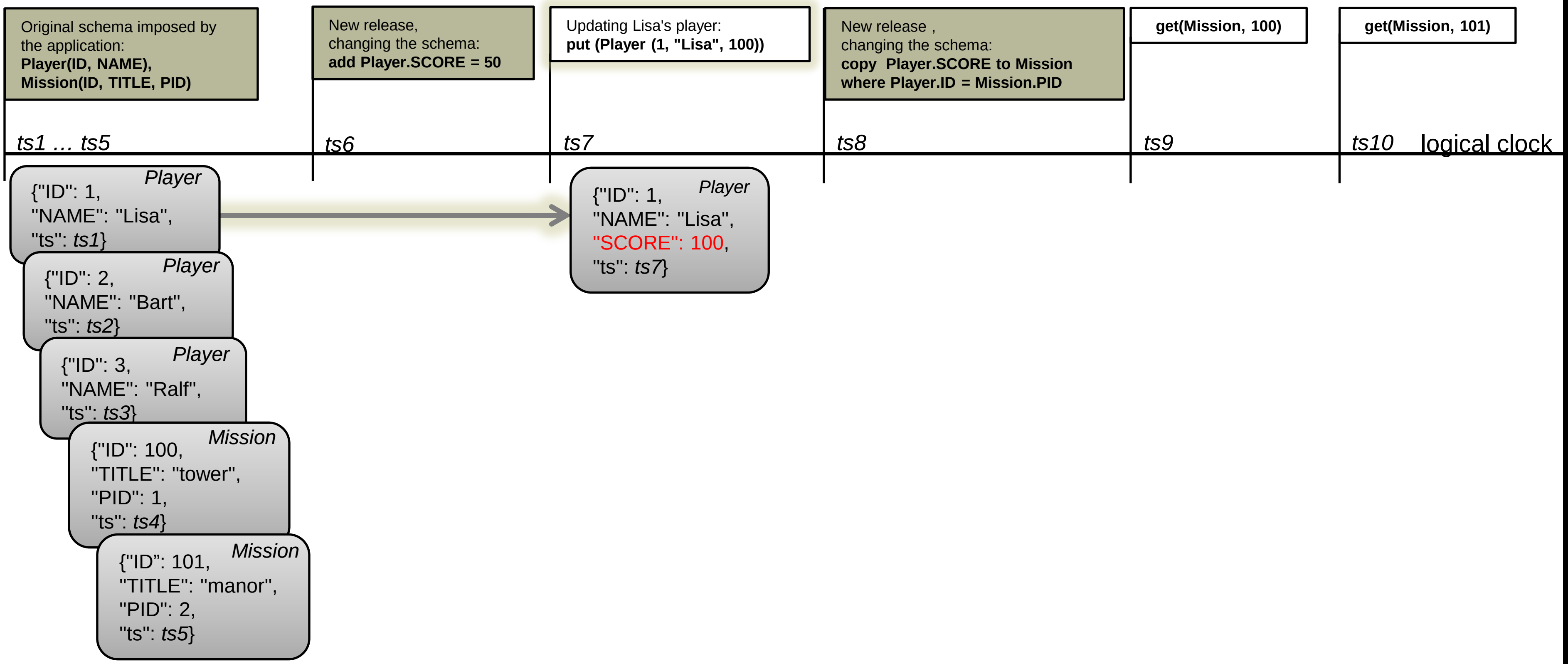
Player
{"ID": 2,
"NAME": "Bart",
"ts": *ts2*}

Player
{"ID": 3,
"NAME": "Ralf",
"ts": *ts3*}

Mission
{"ID": 100,
"TITLE": "tower",
"PID": 1,
"ts": *ts4*}

Mission
{"ID": 101,
"TITLE": "manor",
"PID": 2,
"ts": *ts5*}

LAZY MIGRATION IN DATALUTION



Original schema imposed by the application:
Player(ID, NAME),
Mission(ID, TITLE, PID)

New release,
changing the schema:
add Player.SCORE = 50

Updating Lisa's player:
put (Player (1, "Lisa", 100))

New release ,
changing the schema:
copy Player.SCORE to Mission
where Player.ID = Mission.PID

get(Mission, 100)

get(Mission, 101)

ts1 … ts5 *ts6* *ts7* *ts8* *ts9* *ts10* logical clock

Player
{"ID": 1,
"NAME": "Lisa",
"ts": *ts1*}

Player
{"ID": 2,
"NAME": "Bart",
"ts": *ts2*}

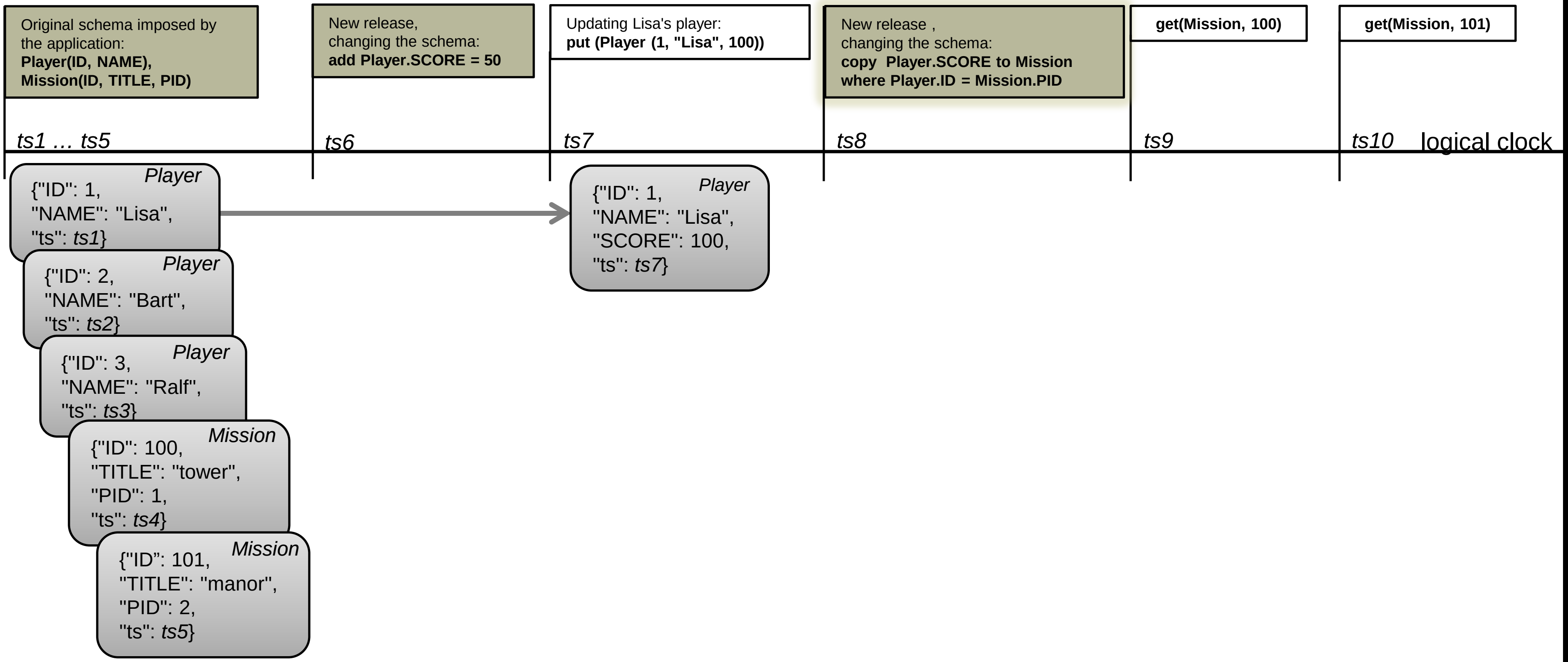
Player
{"ID": 3,
"NAME": "Ralf",
"ts": *ts3*}

Mission
{"ID": 100,
"TITLE": "tower",
"PID": 1,
"ts": *ts4*}

Mission
{"ID": 101,
"TITLE": "manor",
"PID": 2,
"ts": *ts5*}

Player
{"ID": 1,
"NAME": "Lisa",
"SCORE": 100,
"ts": *ts7*}

LAZY MIGRATION IN DATALUTION



Original schema imposed by the application:
Player(ID, NAME),
Mission(ID, TITLE, PID)

New release, changing the schema:
add Player.SCORE = 50

Updating Lisa's player:
put (Player (1, "Lisa", 100))

New release , changing the schema:
copy Player.SCORE to Mission
where Player.ID = Mission.PID

get(Mission, 100)

get(Mission, 101)

ts1 … ts5 *ts6* *ts7* *ts8* *ts9* *ts10* logical clock

Player
{"ID": 1,
"NAME": "Lisa",
"ts": *ts1*}

Player
{"ID": 2,
"NAME": "Bart",
"ts": *ts2*}

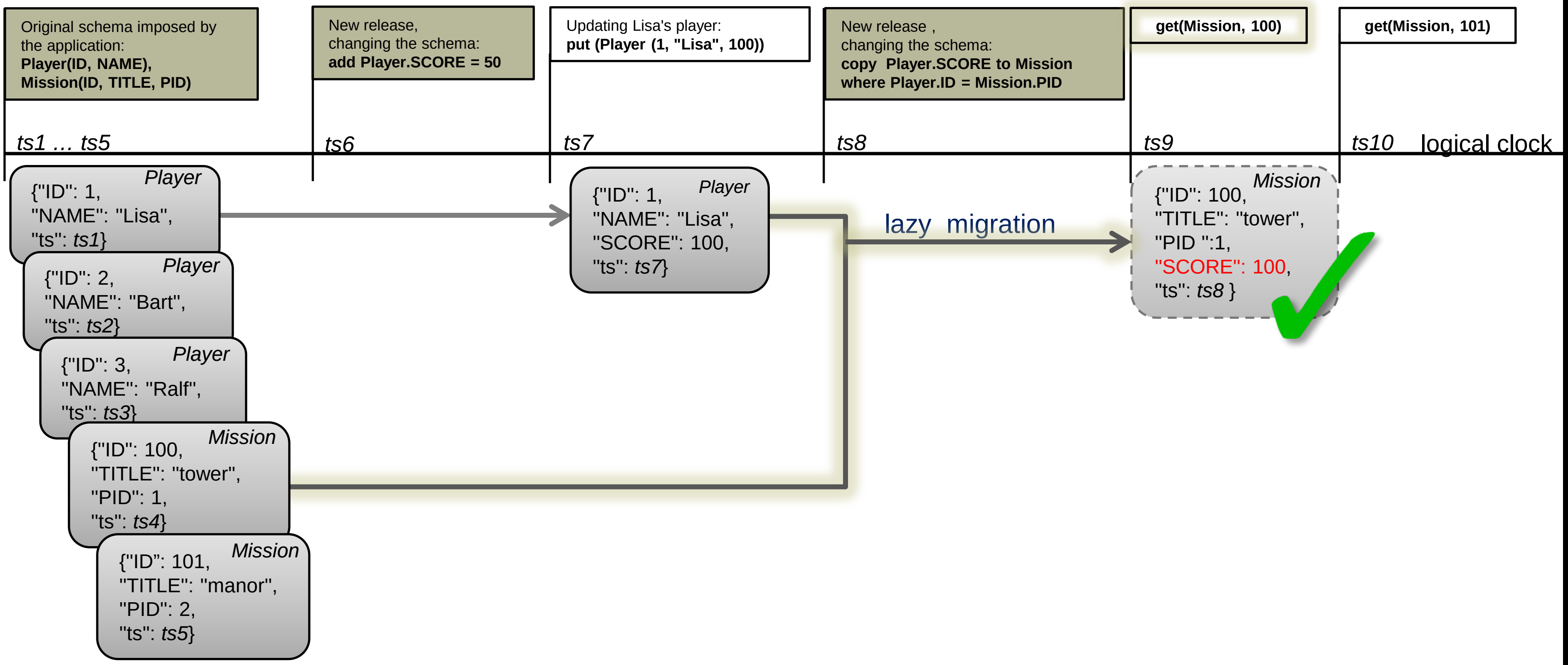
Player
{"ID": 3,
"NAME": "Ralf",
"ts": *ts3*}

Mission
{"ID": 100,
"TITLE": "tower",
"PID": 1,
"ts": *ts4*}

Mission
{"ID": 101,
"TITLE": "manor",
"PID": 2,
"ts": *ts5*}

Player
{"ID": 1,
"NAME": "Lisa",
"SCORE": 100,
"ts": *ts7*}

LAZY MIGRATION IN DATALUTION



Original schema imposed by the application:
Player(ID, NAME),
Mission(ID, TITLE, PID)

New release,
changing the schema:
add Player.SCORE = 50

Updating Lisa's player:
put (Player (1, "Lisa", 100))

New release ,
changing the schema:
copy Player.SCORE to Mission
where Player.ID = Mission.PID

get(Mission, 100)

get(Mission, 101)

ts1 … *ts5* | *ts6* | *ts7* | *ts8* | *ts9* | *ts10* logical clock

Player
{"ID": 1,
"NAME": "Lisa",
"ts": *ts1*}

Player
{"ID": 2,
"NAME": "Bart",
"ts": *ts2*}

Player
{"ID": 3,
"NAME": "Ralf",
"ts": *ts3*}

Mission
{"ID": 100,
"TITLE": "tower",
"PID": 1,
"ts": *ts4*}

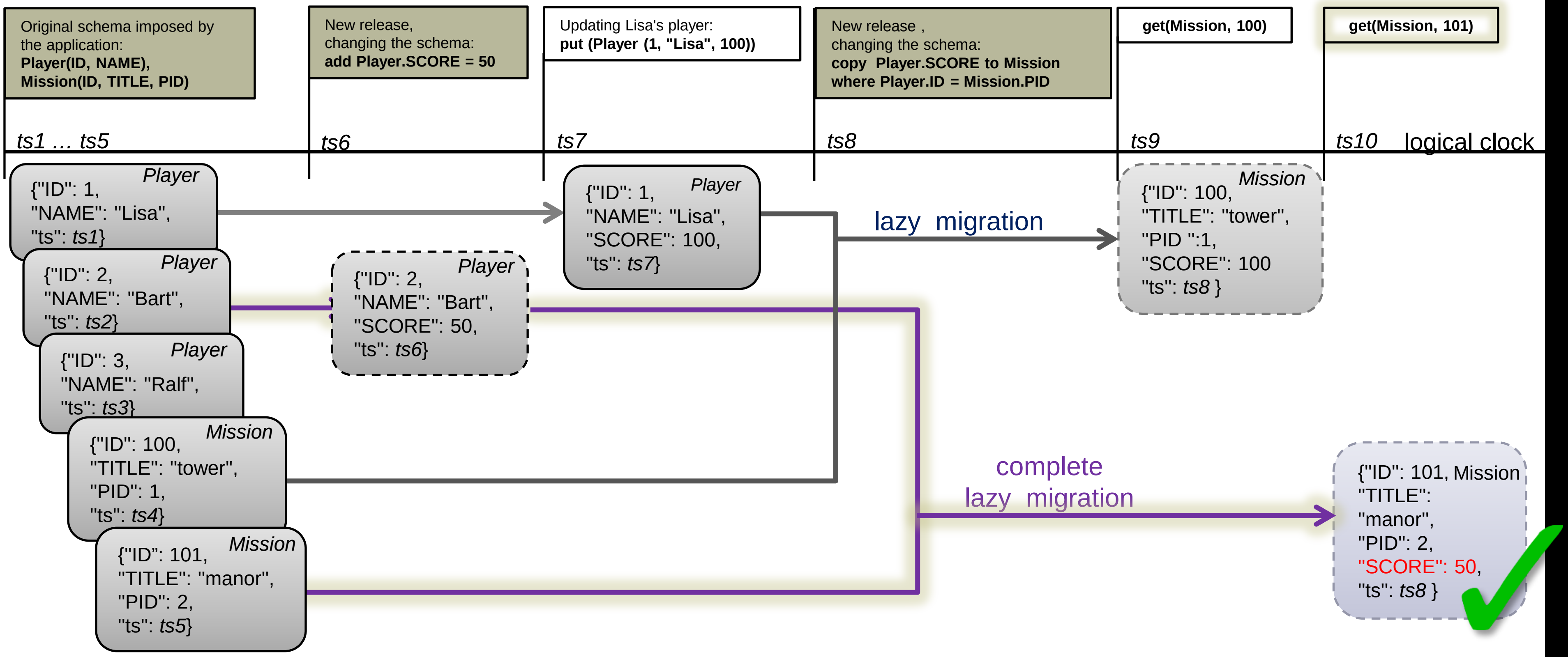
Mission
{"ID": 101,
"TITLE": "manor",
"PID": 2,
"ts": *ts5*}

Player
{"ID": 1,
"NAME": "Lisa",
"SCORE": 100,
"ts": *ts7*}

lazy migration

Mission
{"ID": 100,
"TITLE": "tower",
"PID ":1,
"SCORE": 100,
"ts": *ts8* }

LAZY MIGRATION IN DATALUTION



Original schema imposed by the application:
Player(ID, NAME),
Mission(ID, TITLE, PID)

New release,
changing the schema:
add Player.SCORE = 50

Updating Lisa's player:
put (Player (1, "Lisa", 100))

New release ,
changing the schema:
copy Player.SCORE to Mission
where Player.ID = Mission.PID

get(Mission, 100)

get(Mission, 101)

ts1 … *ts5*   *ts6*   *ts7*   *ts8*   *ts9*   *ts10*   logical clock

Player
{"ID": 1,
"NAME": "Lisa",
"ts": *ts1*}

Player
{"ID": 2,
"NAME": "Bart",
"ts": *ts2*}

Player
{"ID": 3,
"NAME": "Ralf",
"ts": *ts3*}

Mission
{"ID": 100,
"TITLE": "tower",
"PID": 1,
"ts": *ts4*}

Mission
{"ID": 101,
"TITLE": "manor",
"PID": 2,
"ts": *ts5*}

Player
{"ID": 2,
"NAME": "Bart",
"SCORE": 50,
"ts": *ts6*}

Player
{"ID": 1,
"NAME": "Lisa",
"SCORE": 100,
"ts": *ts7*}

lazy migration

Mission
{"ID": 100,
"TITLE": "tower",
"PID ":1,
"SCORE": 100
"ts": *ts8* }

complete
lazy migration

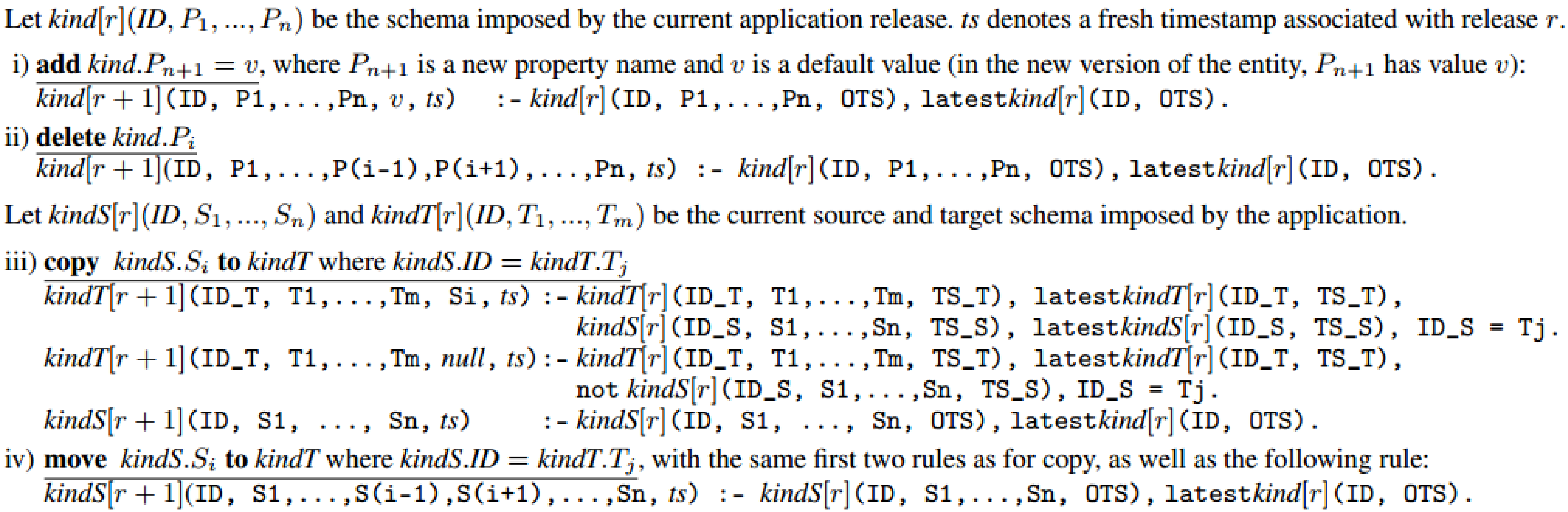
{"ID": 101, Mission
"TITLE":
"manor",
"PID": 2,
"SCORE": 50,
"ts": *ts8* }

✓

DATALOG MODEL (NONRECURSIVE, STRATIFIED)

```
a1: put(Player(1, "Lisa"));
a2: put(Player(1, "Lisa S."));

                r1: Player(1, "Lisa", ts1).
                r2: Player(1, "Lisa S.", ts2).




a3: get("Player", 1);

                r3: legacyPlayer(ID, TS) :-
                      Player(ID, _, TS), Player(ID, _, NTS), TS < NTS.
                r4: latestPlayer(ID, TS) :-
                      Player(ID, _, TS), not legacyPlayer(ID, TS).
                r5: getPlayer(ID, NAME, TS) :-
                      Player(ID, NAME, TS), latestPlayer(ID, TS).
```

transient rule – derived facts not kept around for incremental evaluation



Let $kind[r](ID, P_1, ..., P_n)$ be the schema imposed by the current application release. *ts* denotes a fresh timestamp associated with release r.

i) **add** $kind.P_{n+1} = v$, where P_{n+1} is a new property name and v is a default value (in the new version of the entity, P_{n+1} has value v):
$kind[r+1]$(ID, P1,...,Pn, v, *ts*) :- $kind[r]$(ID, P1,...,Pn, OTS), latest$kind[r]$(ID, OTS).

ii) **delete** $kind.P_i$
$kind[r+1]$(ID, P1,...,P(i-1),P(i+1),...,Pn, *ts*) :- $kind[r]$(ID, P1,...,Pn, OTS), latest$kind[r]$(ID, OTS).

Let $kindS[r](ID, S_1, ..., S_n)$ and $kindT[r](ID, T_1, ..., T_m)$ be the current source and target schema imposed by the application.

iii) **copy** $kindS.S_i$ **to** *kindT* where $kindS.ID = kindT.T_j$
$kindT[r+1]$(ID_T, T1,...,Tm, Si, *ts*) :- $kindT[r]$(ID_T, T1,...,Tm, TS_T), latest$kindT[r]$(ID_T, TS_T),
$kindS[r]$(ID_S, S1,...,Sn, TS_S), latest$kindS[r]$(ID_S, TS_S), ID_S = Tj.
$kindT[r+1]$(ID_T, T1,...,Tm, *null*, *ts*) :- $kindT[r]$(ID_T, T1,...,Tm, TS_T), latest$kindT[r]$(ID_T, TS_T),
not $kindS[r]$(ID_S, S1,...,Sn, TS_S), ID_S = Tj.
$kindS[r+1]$(ID, S1, ..., Sn, *ts*) :- $kindS[r]$(ID, S1, ..., Sn, OTS), latest$kind[r]$(ID, OTS).

iv) **move** $kindS.S_i$ **to** *kindT* where $kindS.ID = kindT.T_j$, with the same first two rules as for copy, as well as the following rule:
$kindS[r+1]$(ID, S1,...,S(i-1),S(i+1),...,Sn, *ts*) :- $kindS[r]$(ID, S1,...,Sn, OTS), latest$kind[r]$(ID, OTS).

DATALUTION: DATALOG-BASED

- **Eager migration: Incremental bottom-up evaluation**
- **Lazy migration: Incremental top-down evaluation**
  - Employing sideways information passing strategies
  - Exploiting uniqueness of identifiers

- **Both strategies always <u>yield the same result</u>**

- **Progress:**
  - Theory in DBPL@SPLASH'15 paper
  - Demo of PoC Datalution at QUDOS'16
  - Ongoing: Integration with NoSQL data store