DATABASE VERSIONING – OUR MOTIVATION

State of the Art: Database Evolution

- App $V_i$
- App $V_{i+1}$
- Evolution Script

Our Goal: Database Versioning

- App $V_i$
- App $V_{i+1}$
- Evolution Script

DATABASE VERSIONING

- CREATE TABLE $R(c_1, ..., c_n)$;
- DROP TABLE $R$;
- ADD COLUMN $c$ AS $f(c_1, ..., c_m)$ INTO $R$;
- DROP COLUMN $c$ FROM $R$;
- DECOMPOSE TABLE $R$ INTO $S(t_1, ..., t_m) \setminus T(t_1, ..., t_m)$;
- (OUTER) JOIN TABLE $R, S$ INTO $T$ WHERE $cond$;
- AGGREGATE $R(g_1, ..., g_n)$ WITH $a_1 = f_1(G, V), ..., a_m = f_m(G, V)$ INTO $S$;
- PARTITION TABLE $R$ INTO $S$ WITH $cond_S \setminus T$ WITH $cond_T$;
- MERGE TABLE $R, S$ INTO $T$;
- RENAME TABLE $R$ INTO $R'$;
- RENAME COLUMN $c$ IN $R$ TO $c'$;


INVERDA – OUR SOLUTION

EXTERNAL SCHEMA

- Each external schema is locally specified with Data Evolution Language (DEL)
- Directed acyclic hyper-graph with Unidirectional Table Cycles as nodes and Mapping (SMDC) as edges.

CONCEPTUAL SCHEMA

- Directed acyclic hyper-graph with Unidirectional Table Cycles as nodes and Mapping (SMDC) as edges.

INTERNAL SCHEMA

- Maintains a valid subset of the mappings.

DATALOG – THE KEY TO INVERTIBILITY

Source

- $getSource$
- $putTarget$

Target

- $getSource$
- $putTarget$

Goal

- $Source = getSource\{putTarget(Source)\}$
- $Target = getTarget\{putTarget(Target)\}$

PARTITION $T$ INTO $R$ WITH $cond_R$, $S$ WITH $cond_S$ DEFAULT $R$;

- $R(p, a) \iff T(p, a), cond(p, c), \sim R^+(p)$
- $S(p, a) \iff T(p, a), cond(p, a), \sim S^+(p), \sim S^{-}(p)$
- $S(p, a) \iff S^{-}(p, a)$
- $T'(p, a) \iff T(p, a), \sim cond(p, a), \sim cond(p, a)$

FURTHER RESEARCH QUESTIONS

- General topics
  - Evolution and versioning of role-based models
  - Explicit query optimization
- External level
  - Versioning model
  - Invariant variants
- Internal level
  - Incremental materialization, Zero downtime
  - Evaluation and advisor for materialization
  - Alternative materialization