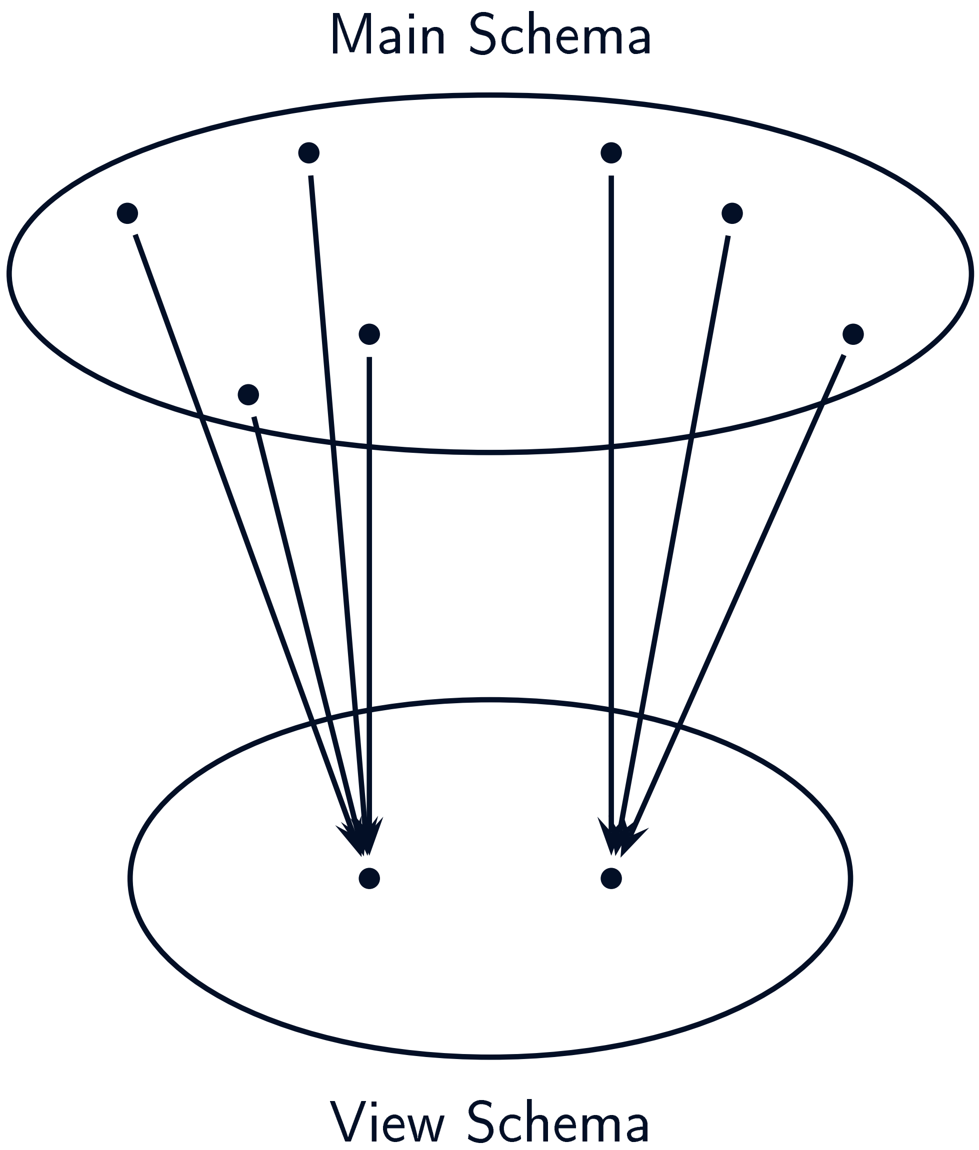
Information-Optimal Reflections of View Updates on Relational Database Schemata

Stephen J. Hegner
Umeå University
Department of Computing Science
Sweden

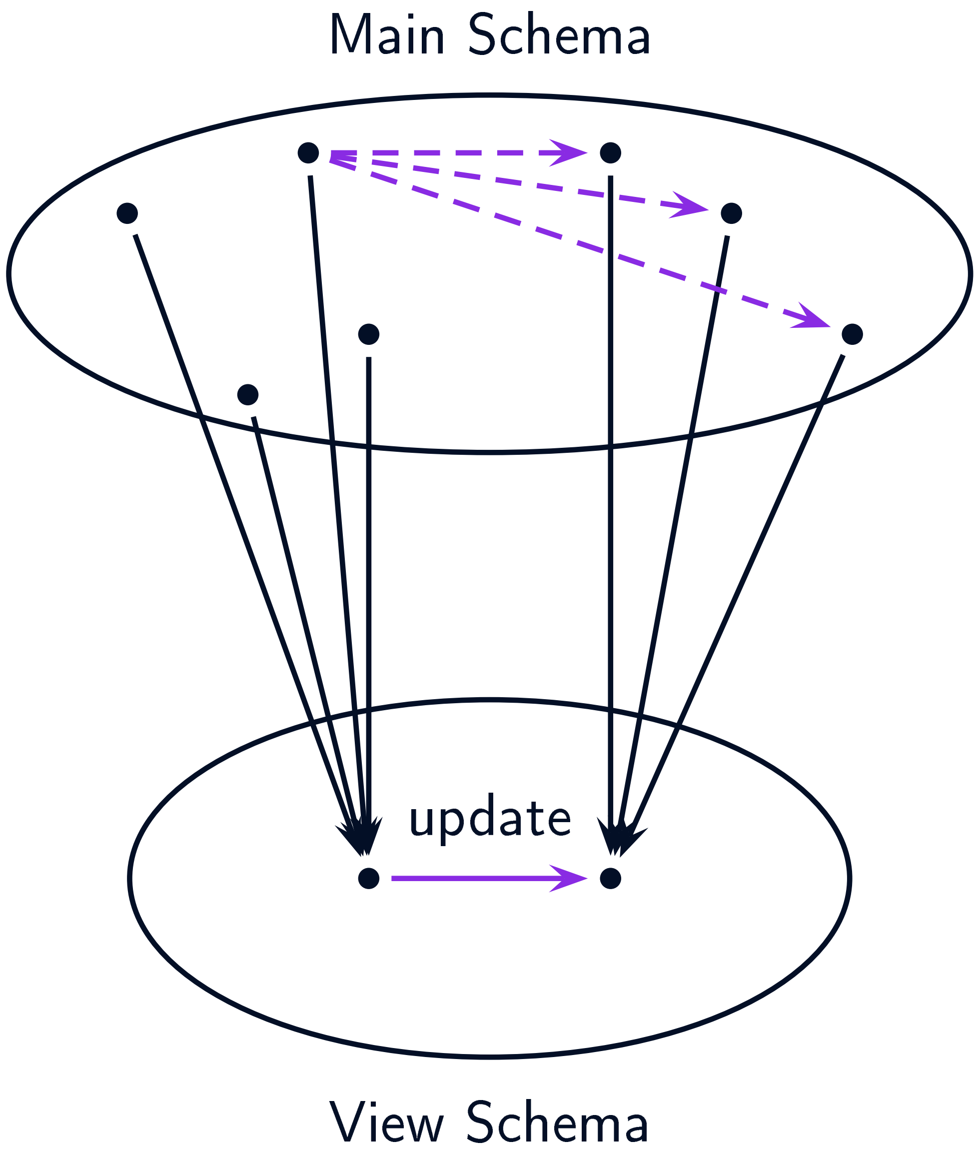
The Update Problem for Database Views

- On the underlying states, the view mapping is generally *surjective* (onto) but not *injective* (one-to-one).



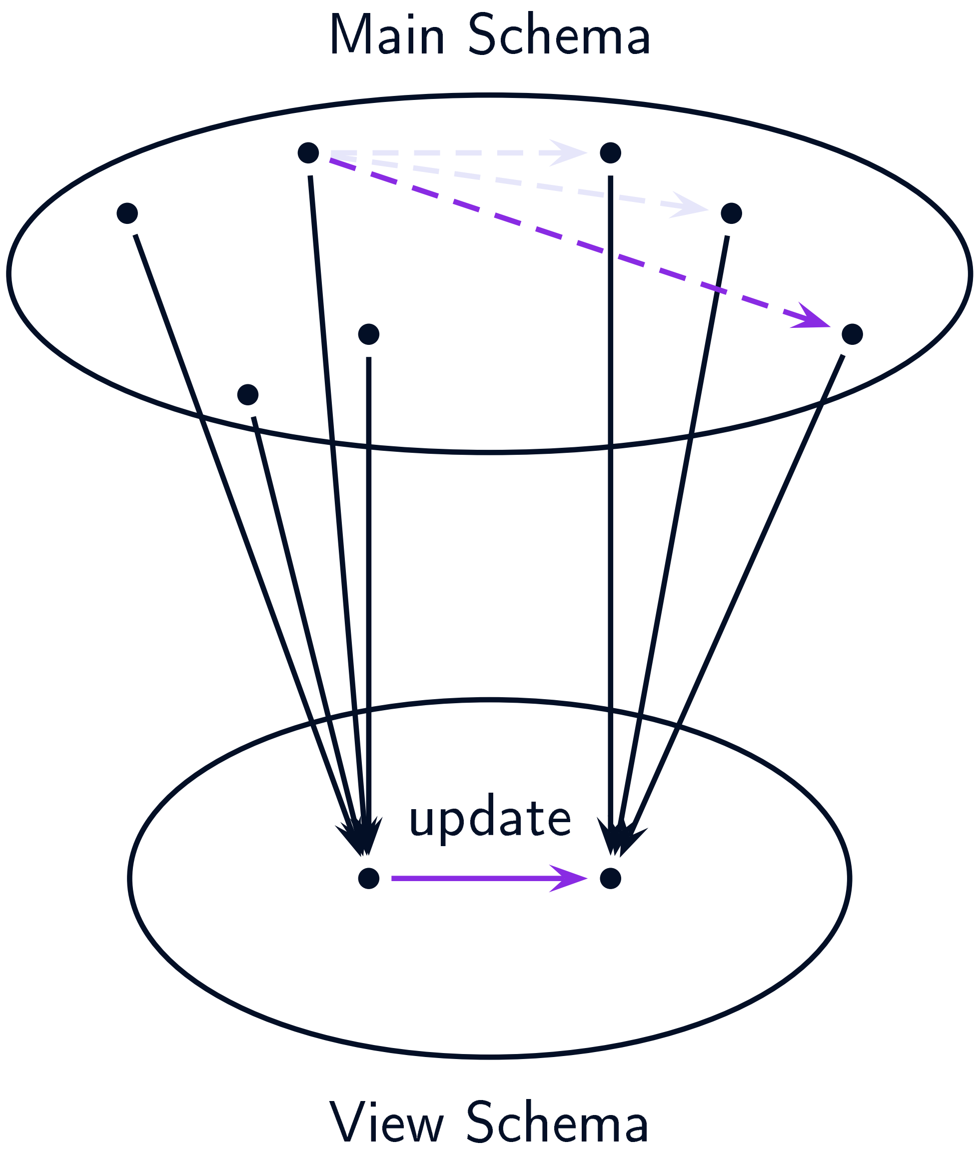
The Update Problem for Database Views

- On the underlying states, the view mapping is generally *surjective* (onto) but not *injective* (one-to-one).
- Thus, a view update has many possible *reflections* to the main schema.



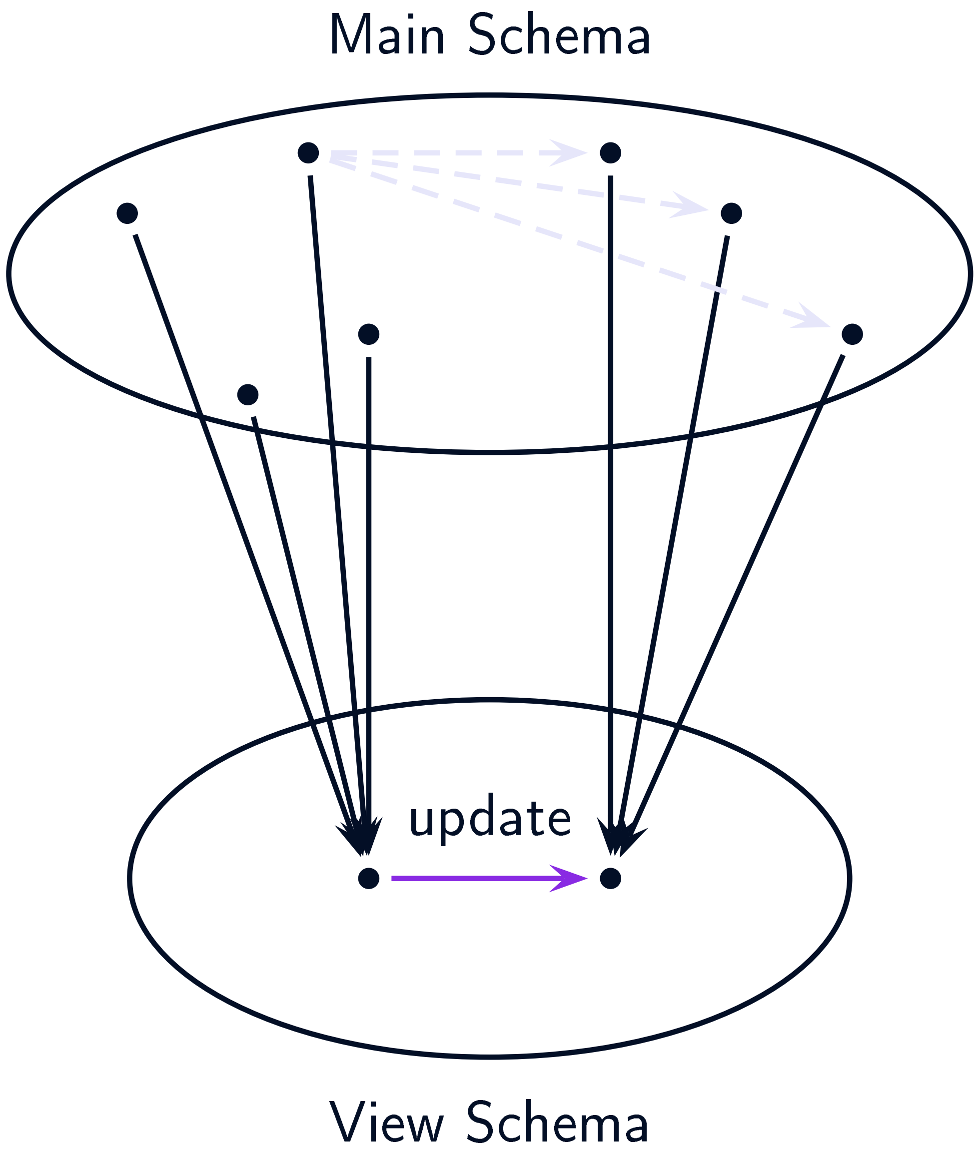
The Update Problem for Database Views

- On the underlying states, the view mapping is generally *surjective* (onto) but not *injective* (one-to-one).
- Thus, a view update has many possible *reflections* to the main schema.
- The problem of identifying a suitable reflection is known as the *update translation problem* or *update reflection problem*.

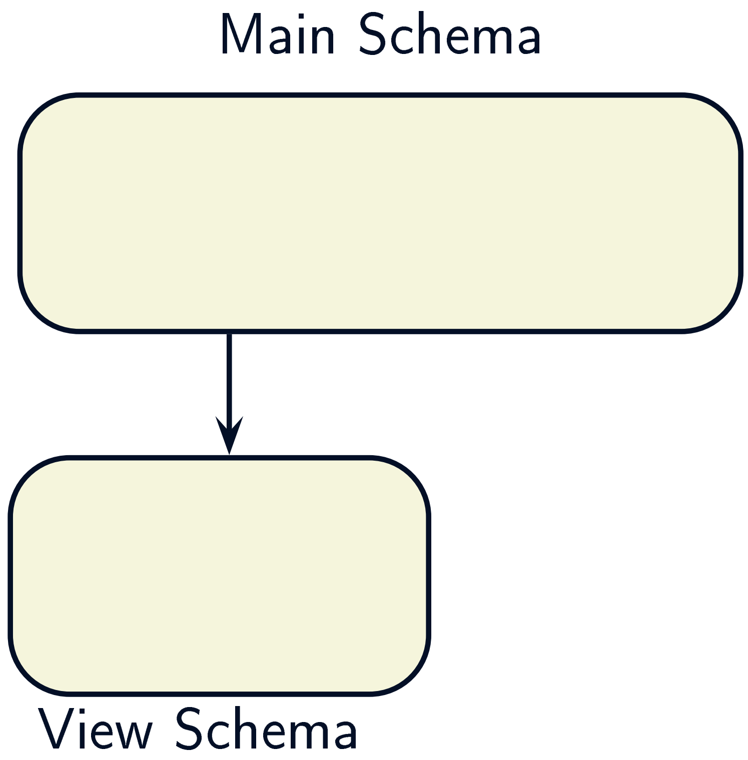


The Update Problem for Database Views

- On the underlying states, the view mapping is generally *surjective* (onto) but not *injective* (one-to-one).
- Thus, a view update has many possible *reflections* to the main schema.
- The problem of identifying a suitable reflection is known as the *update translation problem* or *update reflection problem*.
- With a reasonable definition of suitability, it may not be the case that every view update has a suitable translation.

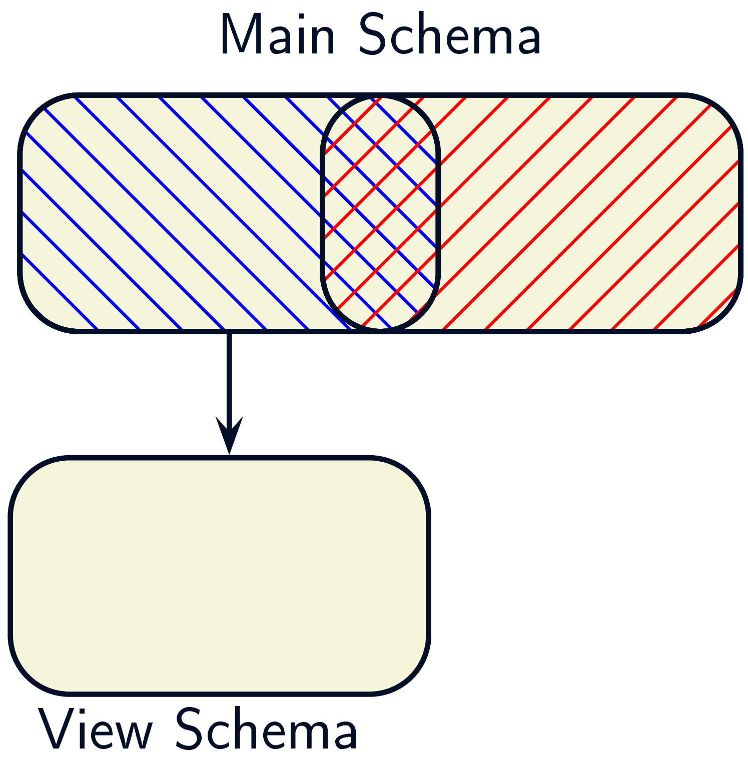


The Gold Standard — the Constant-Complement Strategy



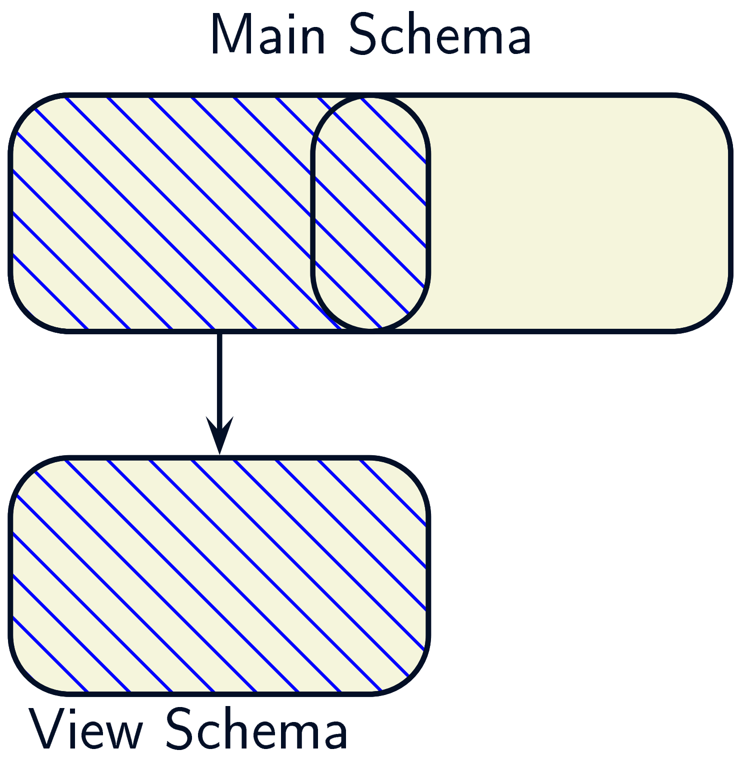
The Gold Standard — the Constant-Complement Strategy

- In the constant-complement strategy [Bancilhon and Spyratos 81], [Hegner 03], the main schema is decomposed into two *meet-complementary* views.



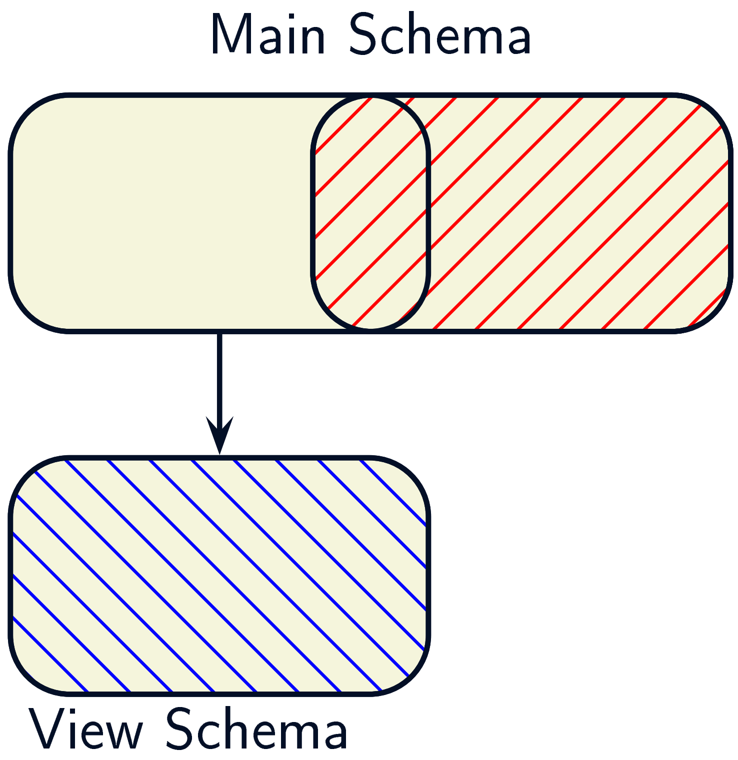
The Gold Standard — the Constant-Complement Strategy

- In the constant-complement strategy [Bancilhon and Spyratos 81], [Hegner 03], the main schema is decomposed into two *meet-complementary* views.
- One is isomorphic to the view schema and tracks its updates exactly.



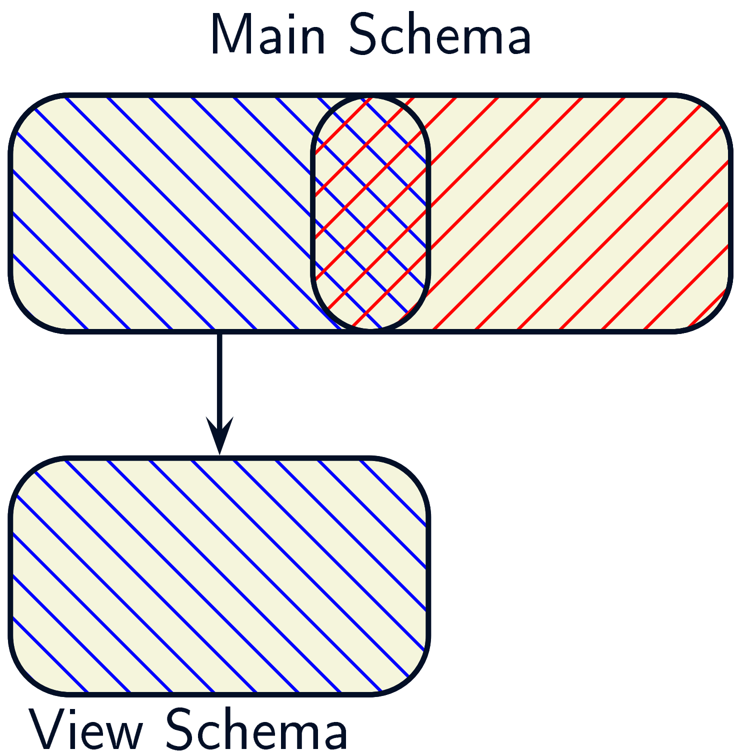
The Gold Standard — the Constant-Complement Strategy

- In the constant-complement strategy [Bancilhon and Spyratos 81], [Hegner 03], the main schema is decomposed into two *meet-complementary* views.
- One is isomorphic to the view schema and tracks its updates exactly.
- The other is held constant for all updates to the view.



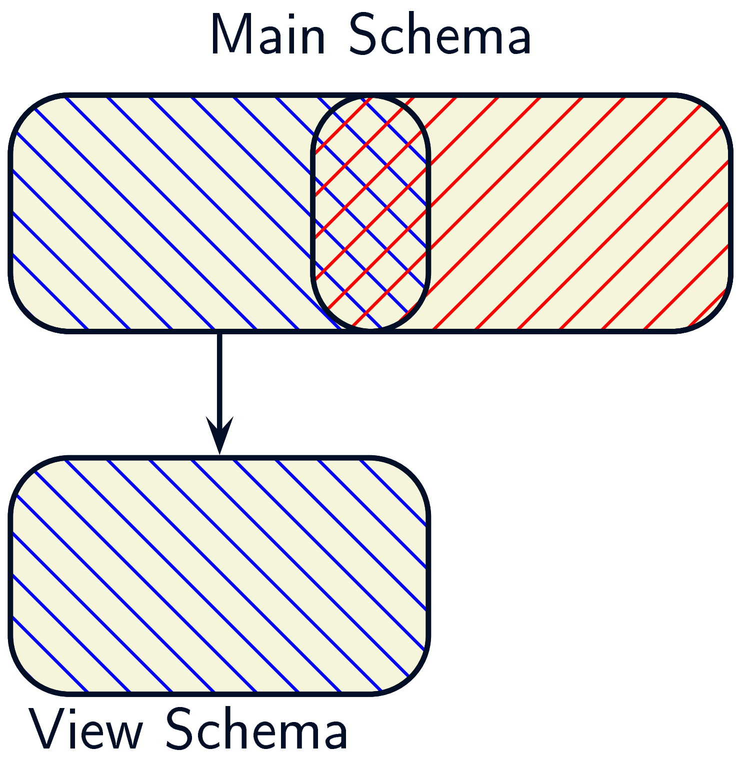
The Gold Standard — the Constant-Complement Strategy

- In the constant-complement strategy [Bancilhon and Spyratos 81], [Hegner 03], the main schema is decomposed into two *meet-complementary* views.
- One is isomorphic to the view schema and tracks its updates exactly.
- The other is held constant for all updates to the view.
- This results in a unique update strategy, although all view updates need not be supported.



The Gold Standard — the Constant-Complement Strategy

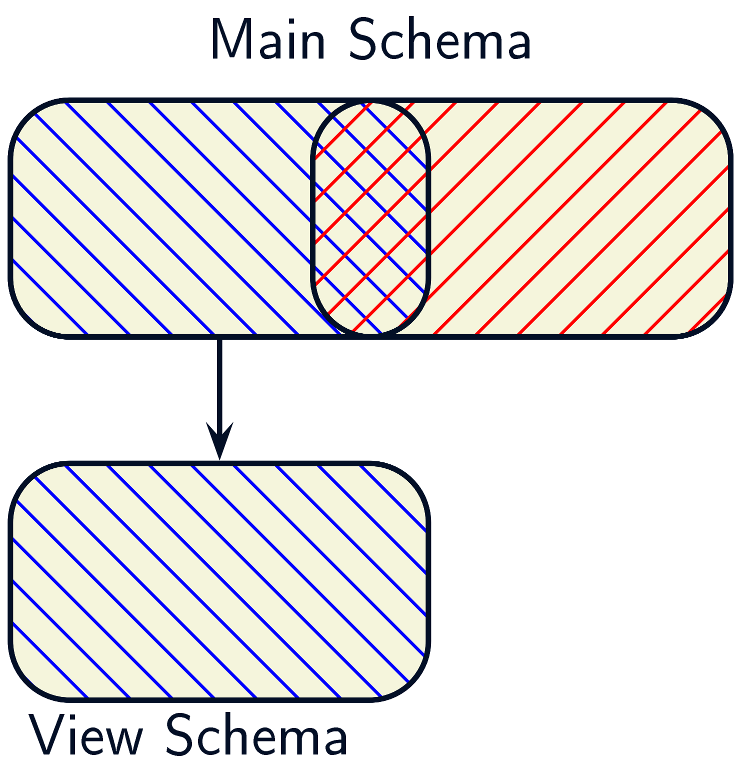
- In the constant-complement strategy [Bancilhon and Spyratos 81], [Hegner 03], the main schema is decomposed into two *meet-complementary* views.
- One is isomorphic to the view schema and tracks its updates exactly.
- The other is held constant for all updates to the view.
- This results in a unique update strategy, although all view updates need not be supported.
- It can be shown [Hegner 03] that this strategy is precisely that which avoids all *update anomalies*.



- Consequently, it is quite limited in the view updates which it allows.

The Gold Standard — the Constant-Complement Strategy

- In the constant-complement strategy [Bancilhon and Spyratos 81], [Hegner 03], the main schema is decomposed into two *meet-complementary* views.
- One is isomorphic to the view schema and tracks its updates exactly.
- The other is held constant for all updates to the view.
- This results in a unique update strategy, although all view updates need not be supported.
- It can be shown [Hegner 03] that this strategy is precisely that which avoids all *update anomalies*.



- Consequently, it is quite limited in the view updates which it allows.
- An example will help illustrate.

An Example of the Constant-Complement Strategy

- Given is the following two-relation main schema.

Main Schema $\mathbf{E}_0$

$R[AB] \bowtie R[BC]$
$R[C] \subseteq S[C]$

$R[ABC]$ $S[CD]$

An Example of the Constant-Complement Strategy

- Given is the following two-relation main schema.

Main Schema $\mathbf{E}_0$

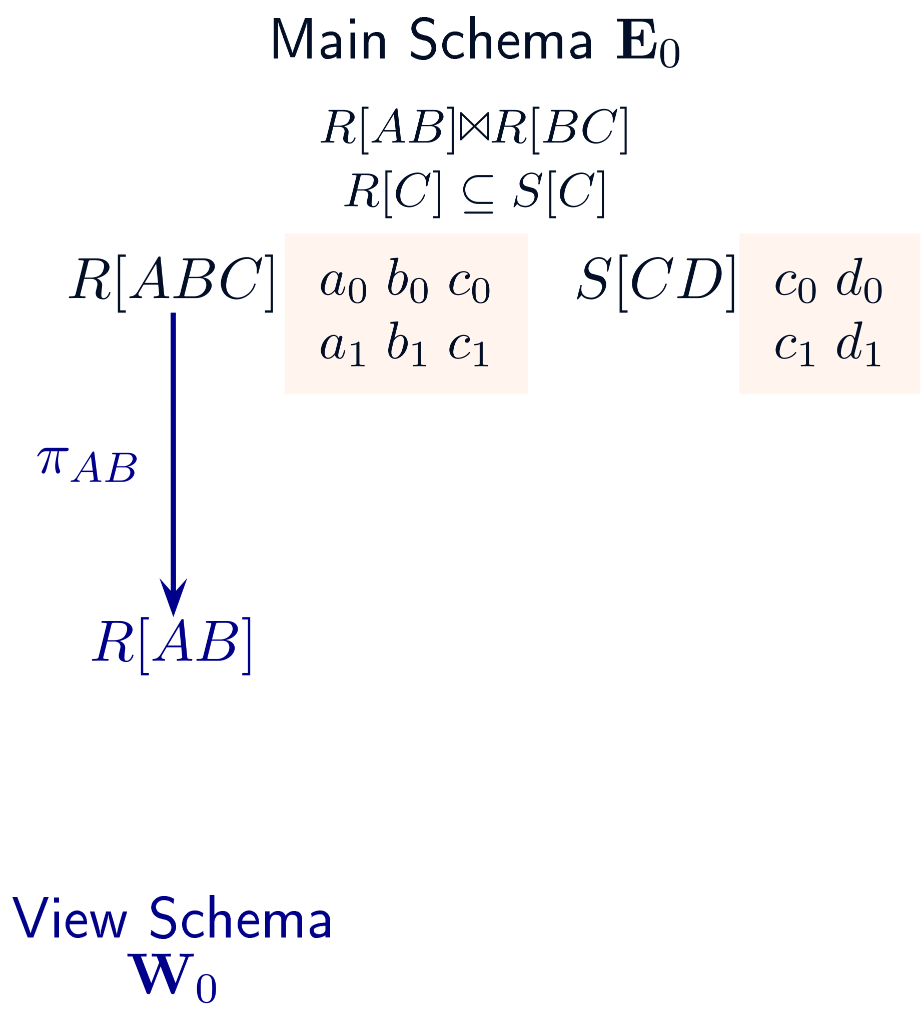
$$R[AB] \bowtie R[BC]$$
$$R[C] \subseteq S[C]$$

$R[ABC]$			
	a_0	b_0	c_0
	a_1	b_1	c_1

$S[CD]$		
	c_0	d_0
	c_1	d_1

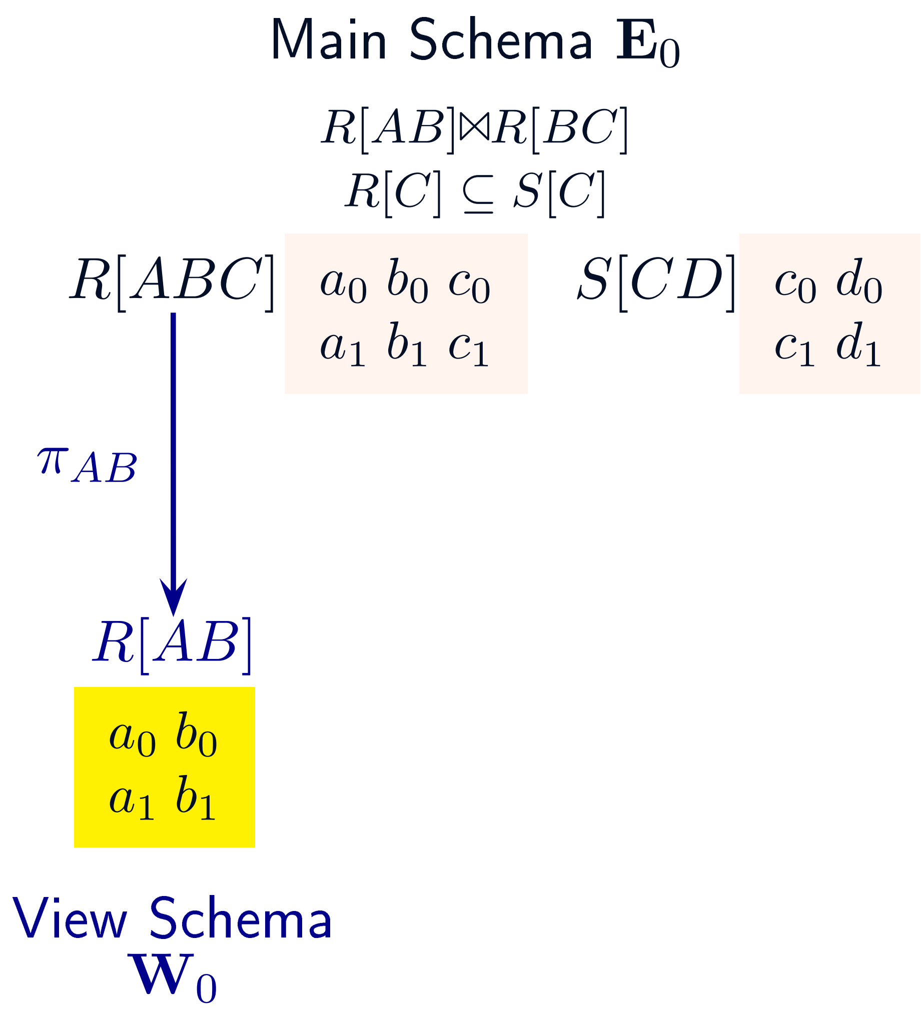
An Example of the Constant-Complement Strategy

- Given is the following two-relation main schema.
- The view schema $\mathbf{W}_0$ to be updated is the AB projection of R.



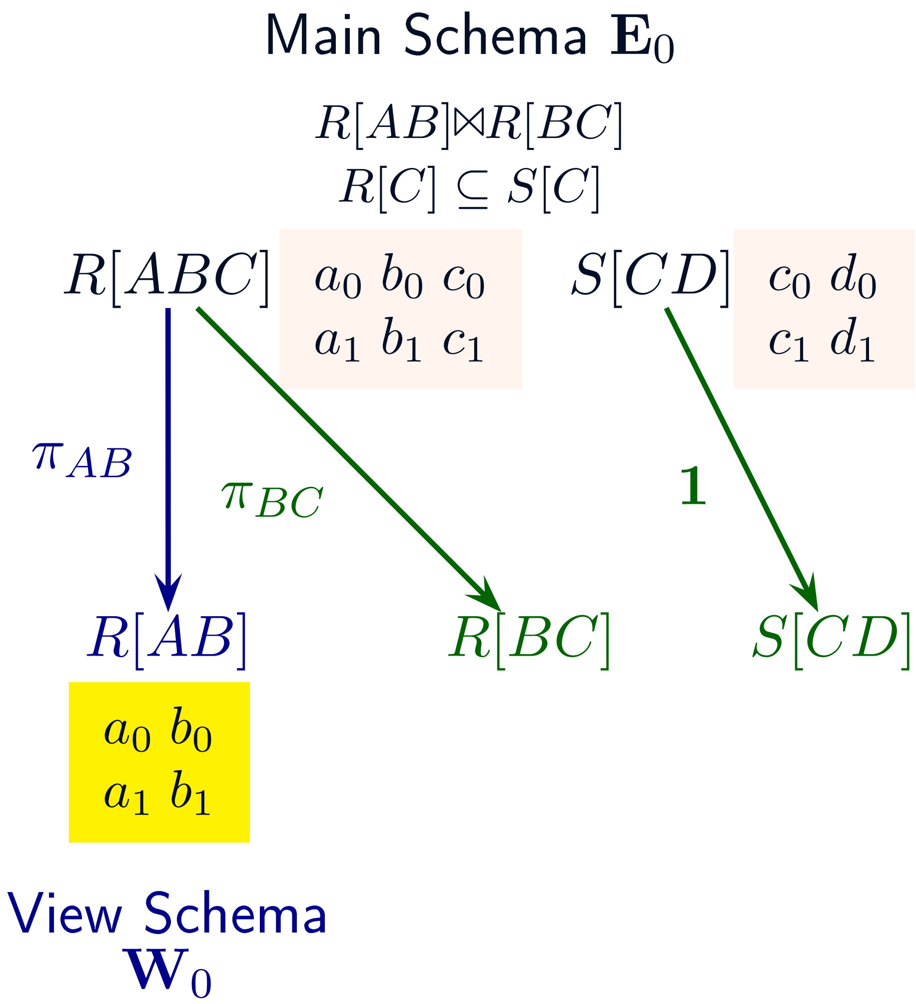
An Example of the Constant-Complement Strategy

- Given is the following two-relation main schema.
- The view schema $\mathbf{W}_0$ to be updated is the AB projection of R.



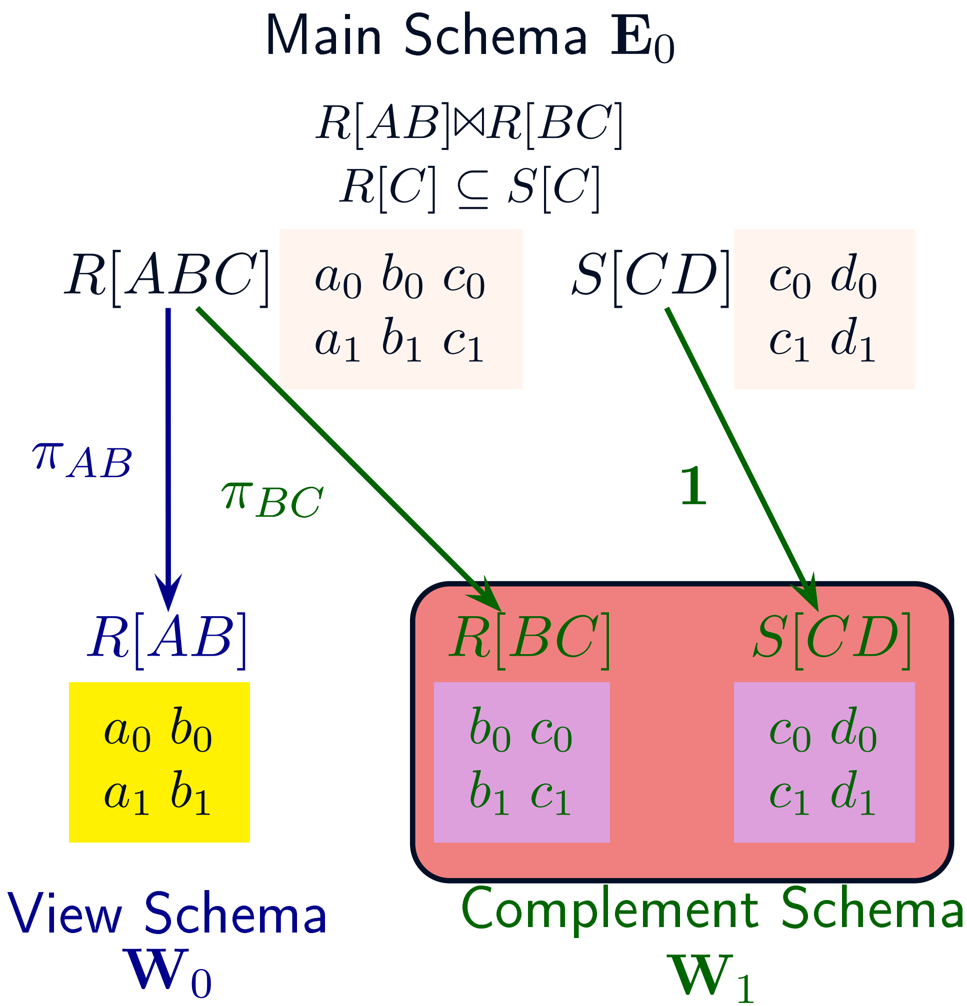
An Example of the Constant-Complement Strategy

- Given is the following two-relation main schema.
- The view schema $\mathbf{W}_0$ to be updated is the AB projection of R.
- The *natural complement* $\mathbf{W}_1$ consists of the BC projection of R and all of S.



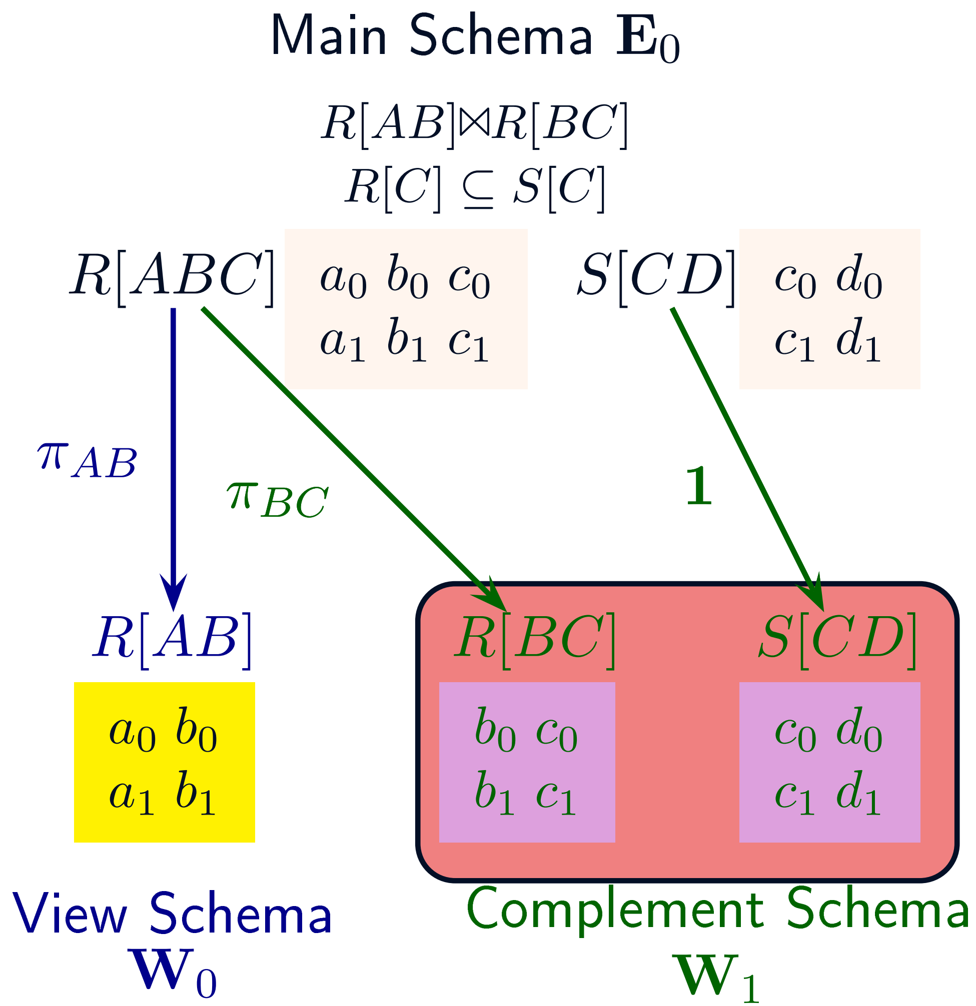
An Example of the Constant-Complement Strategy

- Given is the following two-relation main schema.
- The view schema $\mathbf{W}_0$ to be updated is the AB projection of R.
- The *natural complement* $\mathbf{W}_1$ consists of the BC projection of R and all of S.
- With $\mathbf{W}_1$ constant, the allowable updates to the view are precisely those which keep the *meet* $R[B]$ constant.



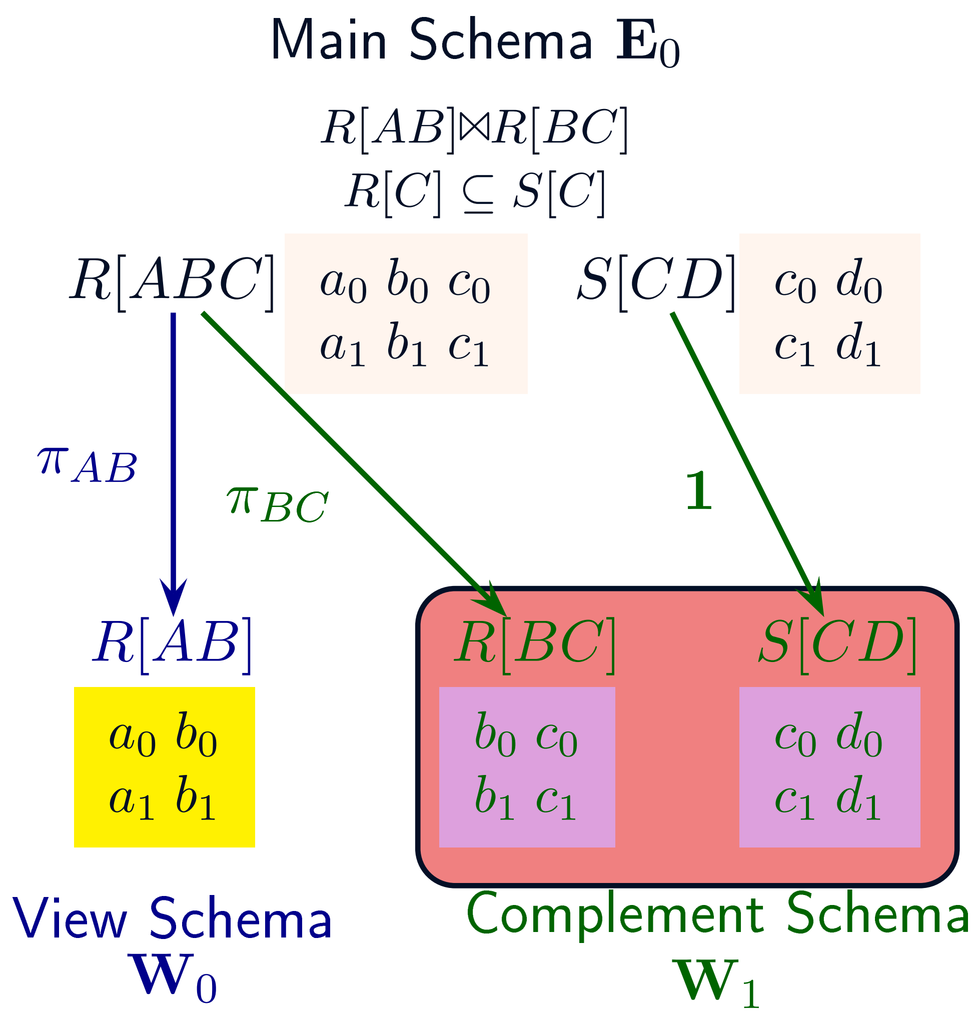
An Example of the Constant-Complement Strategy

- Given is the following two-relation main schema.
- The view schema $\mathbf{W}_0$ to be updated is the AB projection of R.
- The *natural complement* $\mathbf{W}_1$ consists of the BC projection of R and all of S.
- With $\mathbf{W}_1$ constant, the allowable updates to the view are precisely those which keep the *meet* $R[B]$ constant.
- In particular:
    - Deletion of (a_1, b_1) is not allowed.
    - Insertion of (a_2, b_2) is not allowed.



An Example of the Constant-Complement Strategy

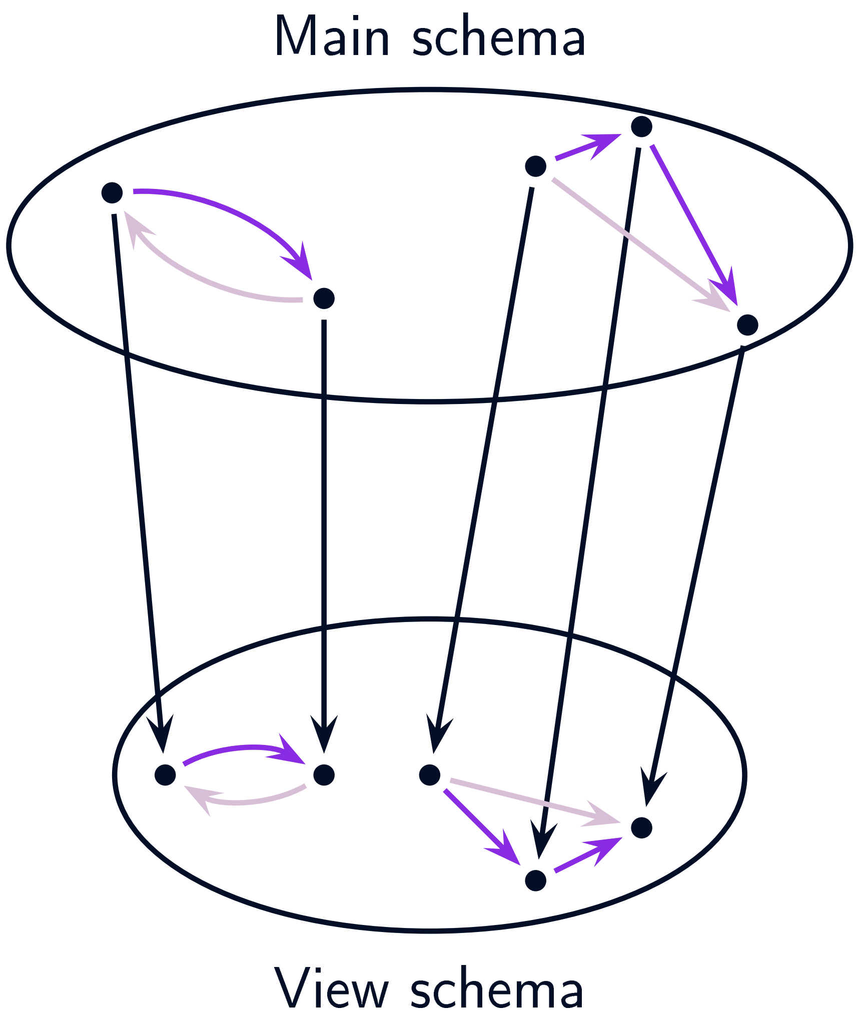
- Given is the following two-relation main schema.
- The view schema $\mathbf{W}_0$ to be updated is the AB projection of R.
- The *natural complement* $\mathbf{W}_1$ consists of the BC projection of R and all of S.
- With $\mathbf{W}_1$ constant, the allowable updates to the view are precisely those which keep the *meet* $R[B]$ constant.
- In particular:
  - Deletion of (a_1, b_1) is not allowed.
  - Insertion of (a_2, b_2) is not allowed.



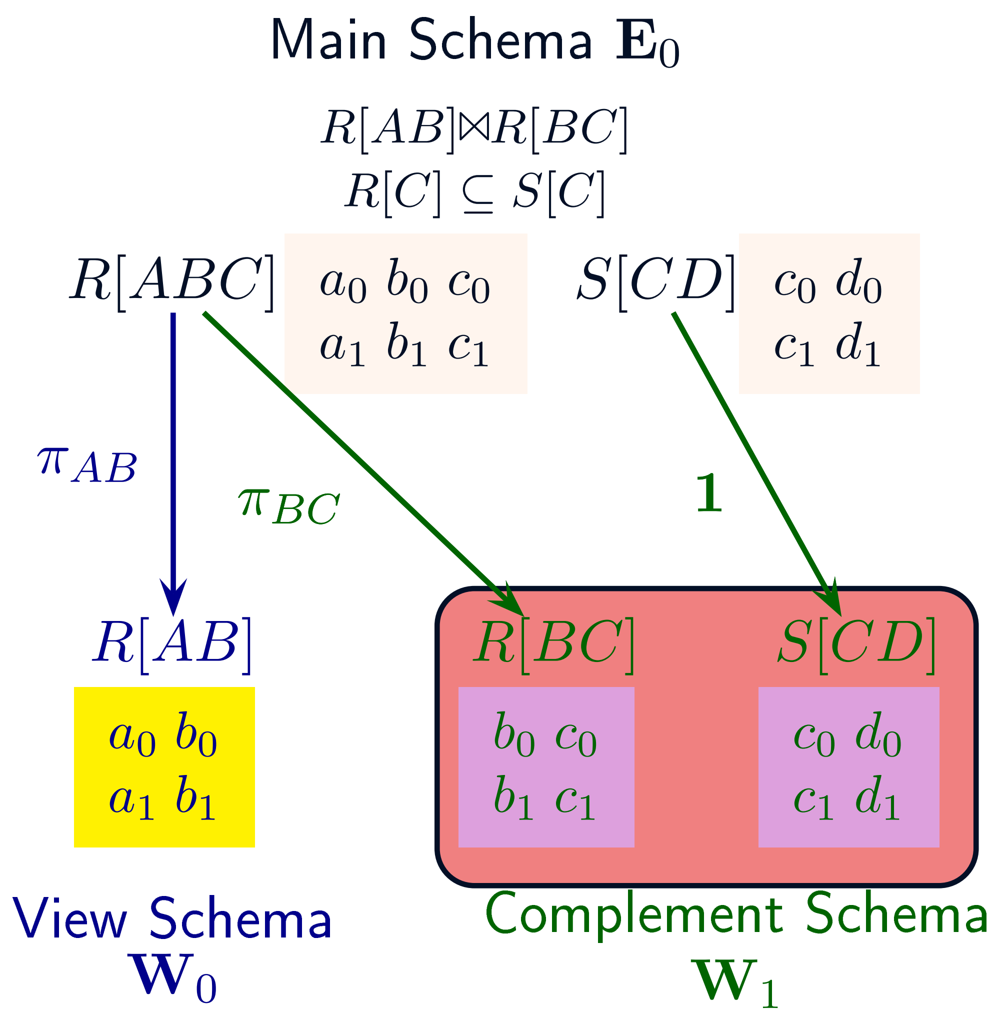
- On the other hand, conceptually, constant-complement view update avoids all *update anomalies*.

Characterization of Admissible View Updates under Constant Complement

- The critical features of constant complement update reflections: *reversibility*, *transitivity*, and *reflection of updates*.

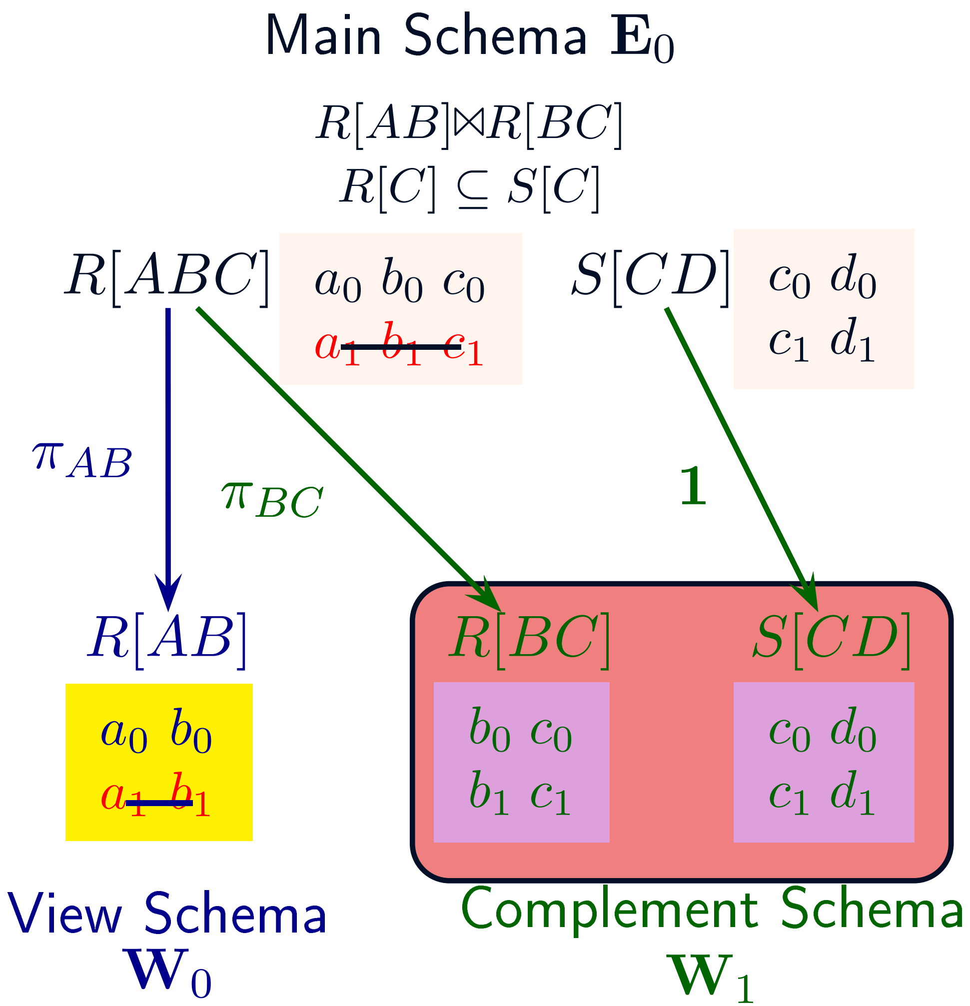


- The critical features of constant complement update reflections: *reversibility*, *transitivity*, and *reflection of updates*.
- Examples of non-admissibility:
    - Deletion of (a_1, b_1) violates reversibility.



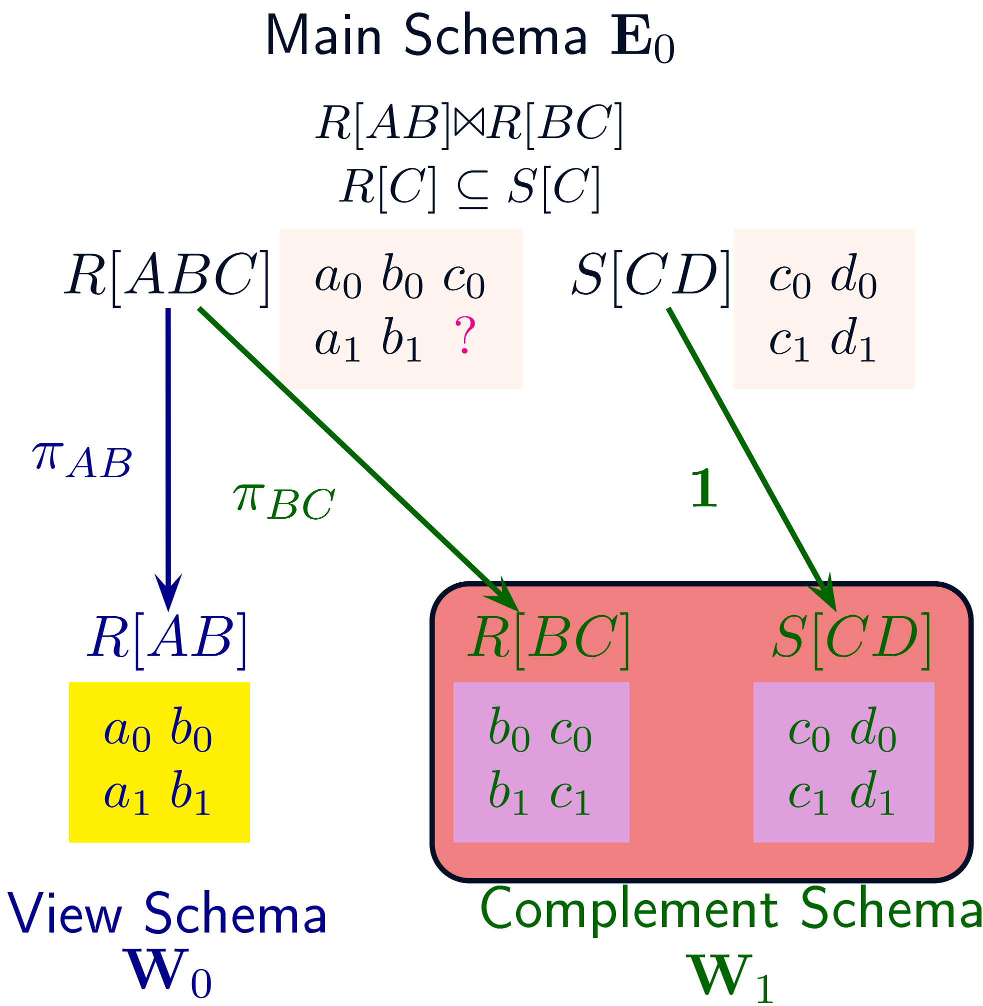
Characterization of Admissible View Updates under Constant Complement

- The critical features of constant complement update reflections: *reversibility*, *transitivity*, and *reflection of updates*.
- Examples of non-admissibility:
    - Deletion of (a_1, b_1) violates reversibility.



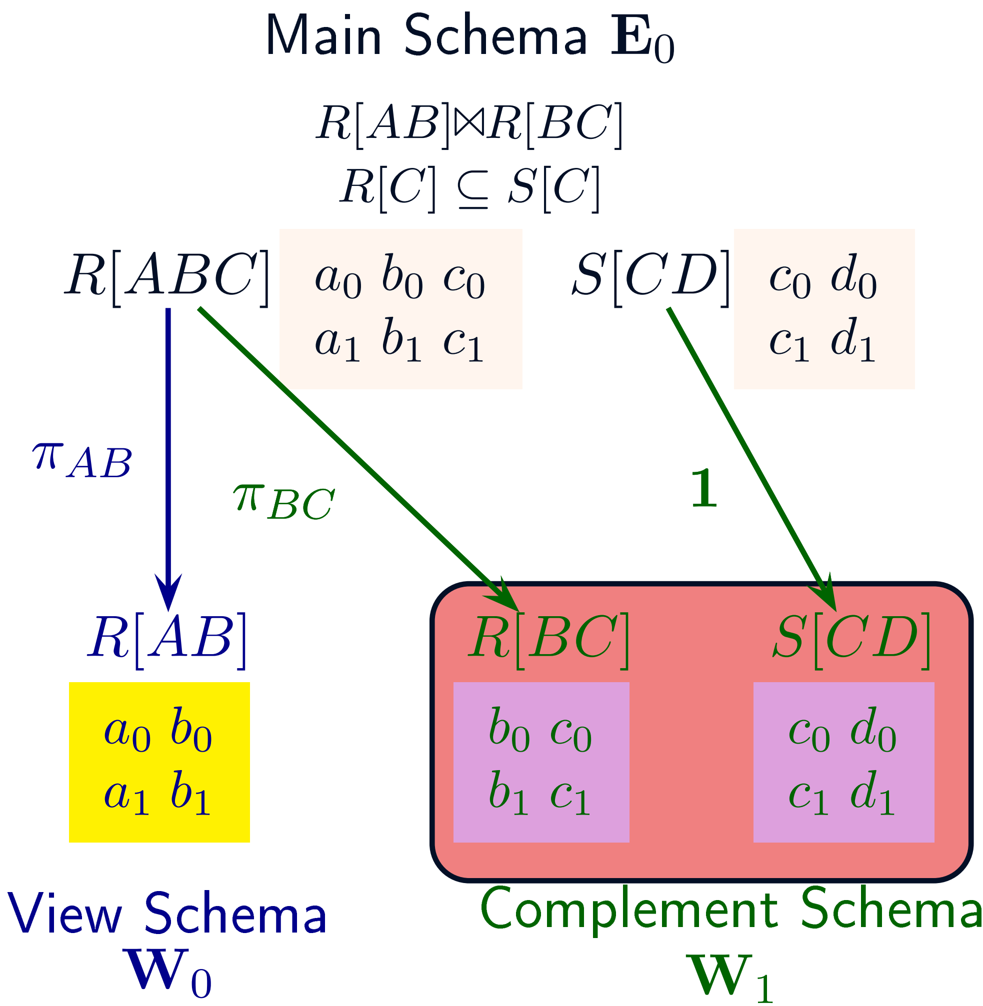
Characterization of Admissible View Updates under Constant Complement

- The critical features of constant complement update reflections: *reversibility*, *transitivity*, and *reflection of updates*.
- Examples of non-admissibility:
  - Deletion of (a_1, b_1) violates reversibility.



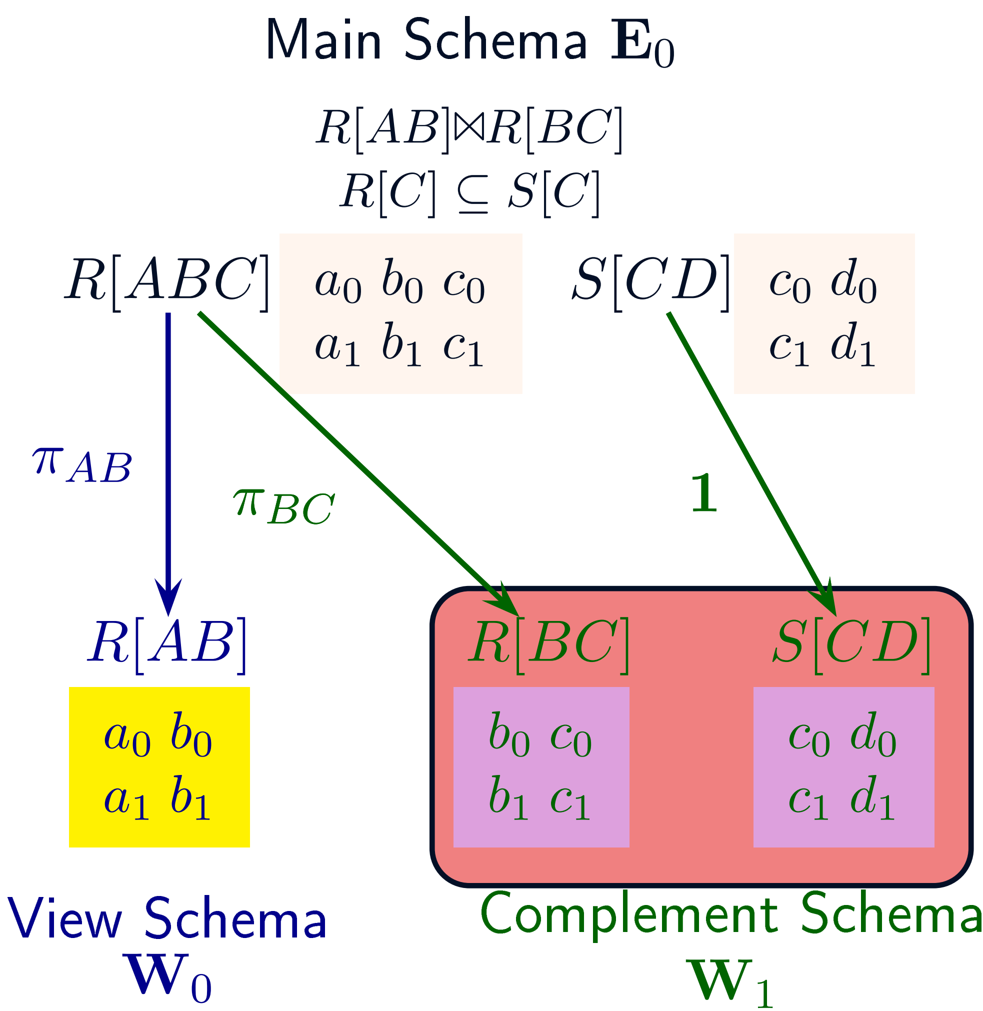
Characterization of Admissible View Updates under Constant Complement

- The critical features of constant complement update reflections: *reversibility*, *transitivity*, and *reflection of updates*.
- Examples of non-admissibility:
  - Deletion of (a_1, b_1) violates reversibility.
  - Insertion of (a_2, b_2) must match its subsequent deletion, which fails for the reason above.

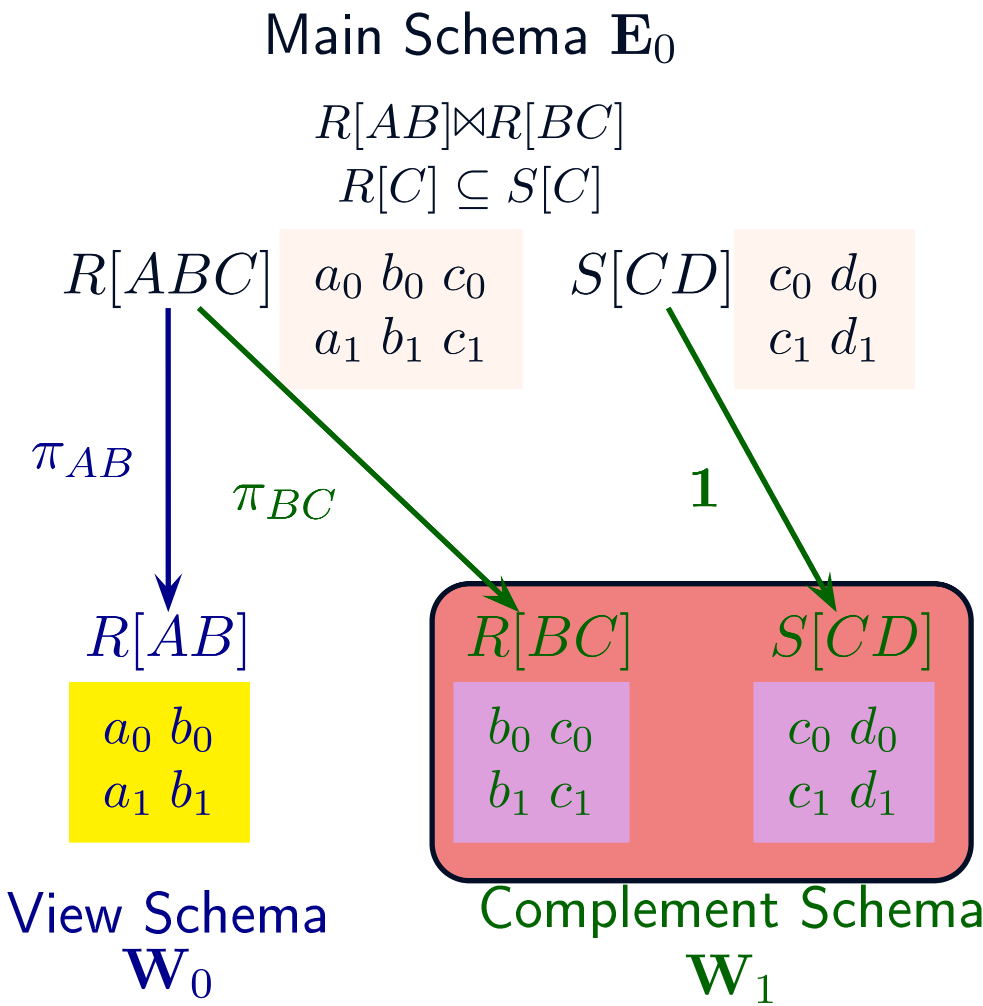


Characterization of Admissible View Updates under Constant Complement

- The critical features of constant complement update reflections: *reversibility*, *transitivity*, and *reflection of updates*.
- Examples of non-admissibility:
  - Deletion of (a_1, b_1) violates reversibility.
  - Insertion of (a_2, b_2) must match its subsequent deletion, which fails for the reason above.
  - Examples which satisfy reversibility but violate transitivity exist as well, but are more complex.



- The critical features of constant complement update reflections: *reversibility*, *transitivity*, and *reflection of updates*.
- Examples of non-admissibility:
    - Deletion of (a_1, b_1) violates reversibility.
    - Insertion of (a_2, b_2) must match its subsequent deletion, which fails for the reason above.
    - Examples which satisfy reversibility but violate transitivity exist as well, but are more complex.



- *Bottom line*: The price of avoiding update anomalies completely is very high.

Extending the Constant-Complement Strategy

- There are two principal approaches to extending the constant-complement strategy:

Extending the Constant-Complement Strategy

- There are two principal approaches to extending the constant-complement strategy:
  - Limited scope: automated decision or decision by one user:

Extending the Constant-Complement Strategy

- There are two principal approaches to extending the constant-complement strategy:

  - Limited scope: automated decision or decision by one user:
    - *Ranked preference* of reflections to the main schema, usually based upon minimization of change.

Extending the Constant-Complement Strategy

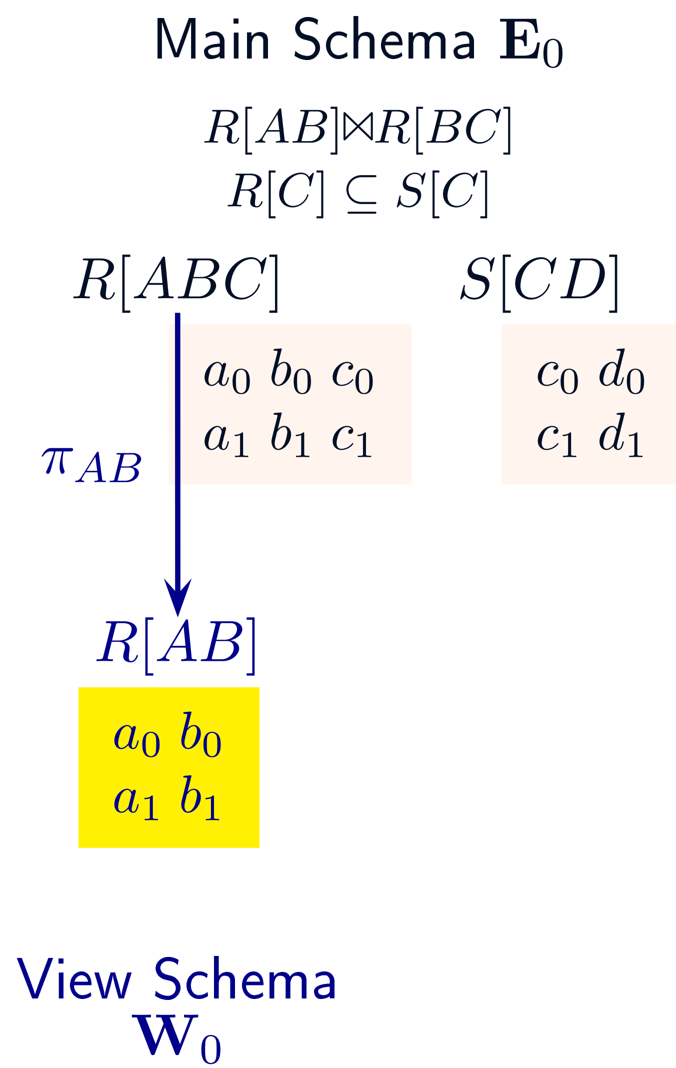
- There are two principal approaches to extending the constant-complement strategy:
  - Limited scope: automated decision or decision by one user:
    - *Ranked preference* of reflections to the main schema, usually based upon minimization of change.
  - Broad scope: decision via the *cooperation* of many users. [Hegner & Schmidt, ADBIS 2007]
    - The complement is updated in a *negotiation* with other users.
    - The complement may in fact be represented as an interconnection of smaller views — *database components*.

Extending the Constant-Complement Strategy

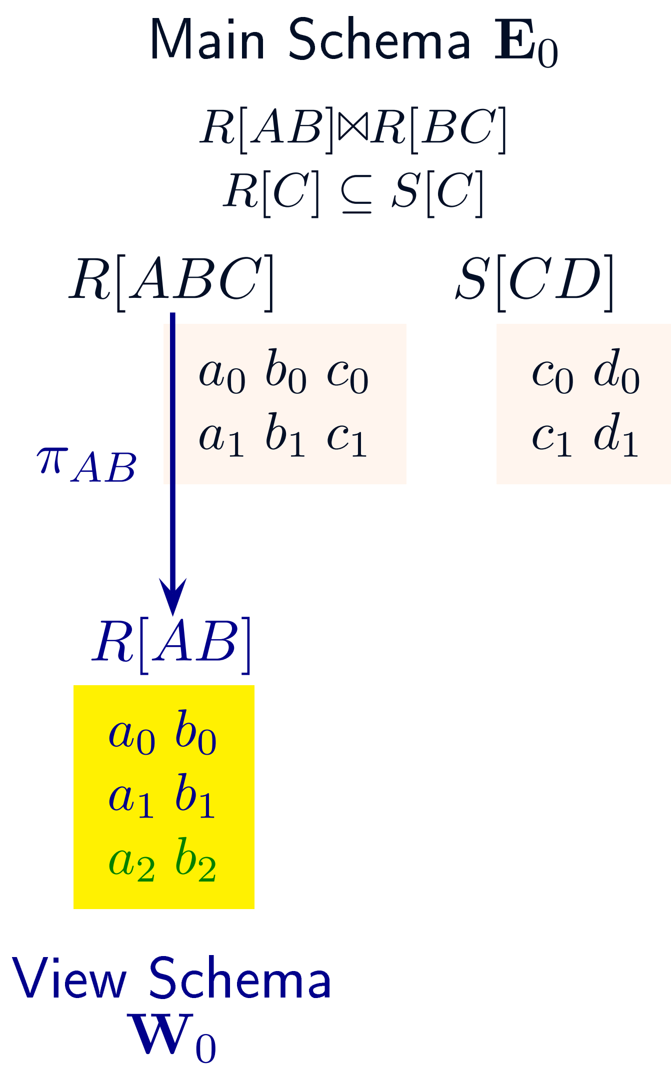
- There are two principal approaches to extending the constant-complement strategy:
  - Limited scope: automated decision or decision by one user:
    - *Ranked preference* of reflections to the main schema, usually based upon minimization of change.
  - Broad scope: decision via the *cooperation* of many users. [Hegner & Schmidt, ADBIS 2007]
    - The complement is updated in a *negotiation* with other users.
    - The complement may in fact be represented as an interconnection of smaller views — *database components*.
  - In this work, the *limited scope* approach, via minimization of change is investigated.

The Idea of Minimal Change

- Consider the update Insert$\langle R(a_2, b_2)\rangle$ into $\mathbf{W}_1$.

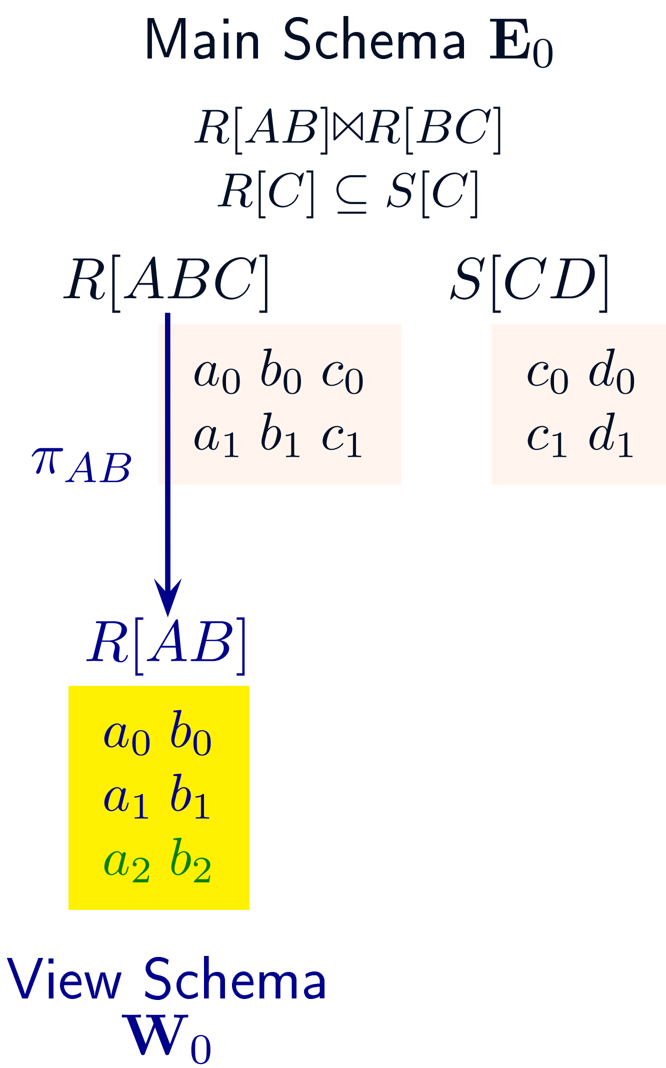


- Consider the update Insert$\langle R(a_2, b_2)\rangle$ into $\mathbf{W}_1$.
- The following alternatives for the reflection are all *tuple minimal* — no proper subset is a solution.
    - Insert$\langle R(a_2, b_2, c_2), S(c_2, d_2)\rangle$
    - Insert$\langle R(a_2, b_2, c_3), S(c_3, d_3)\rangle$
    - Insert$\langle R(a_2, b_2, c_2), S(c_2, d_1)\rangle$
    - Insert$\langle R(a_2, b_2, c_0)\rangle$



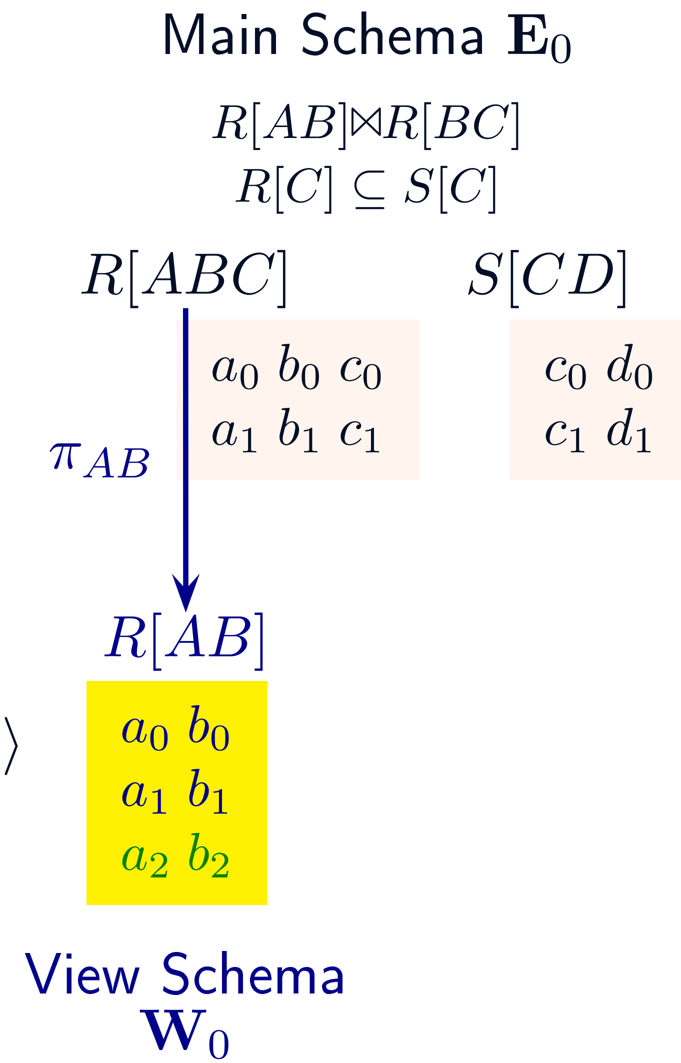
The Idea of Minimal Change

- Consider the update Insert$\langle R(a_2, b_2)\rangle$ into $\mathbf{W}_1$.
- The following alternatives for the reflection are all *tuple minimal* — no proper subset is a solution.
  - Insert$\langle R(a_2, b_2, c_2), S(c_2, d_2)\rangle$
  - Insert$\langle R(a_2, b_2, c_3), S(c_3, d_3)\rangle$
  - Insert$\langle R(a_2, b_2, c_2), S(c_2, d_1)\rangle$
  - Insert$\langle R(a_2, b_2, c_0)\rangle$
- The following alternative is not tuple minimal.
  - Insert$\langle R(a_2, b_2, c_2), S(c_2, d_2), R(a_2, b_2, c_3), S(c_3, d_3)\rangle$



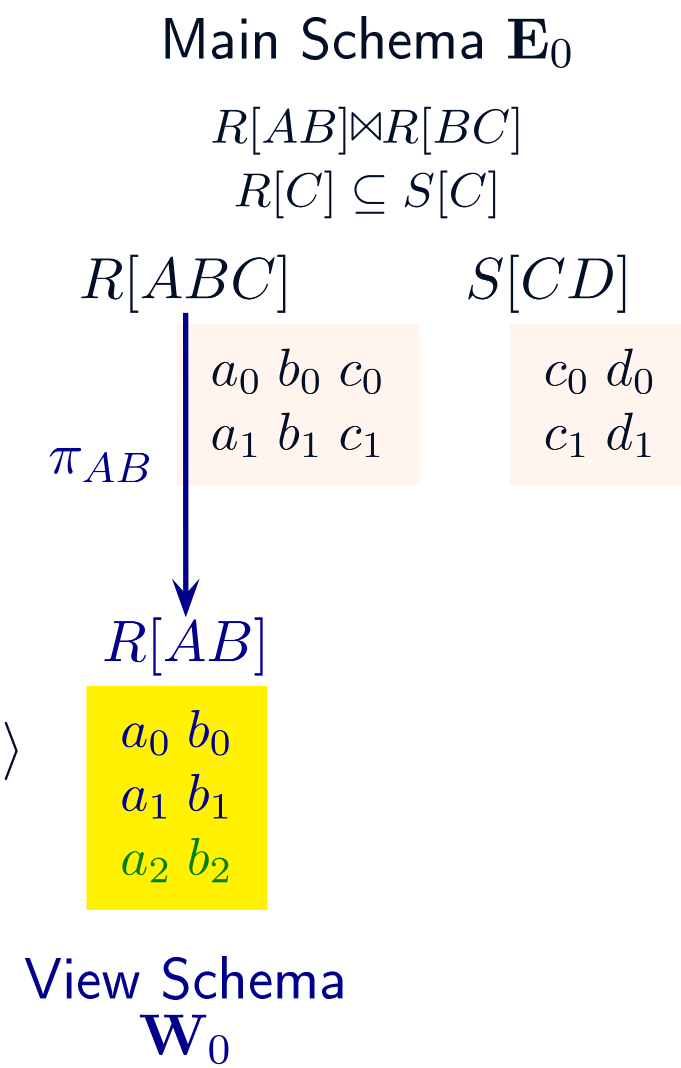
The Idea of Minimal Change

- Consider the update Insert$\langle R(a_2, b_2)\rangle$ into $\mathbf{W}_1$.
- The following alternatives for the reflection are all *tuple minimal* — no proper subset is a solution.
  - Insert$\langle R(a_2, b_2, c_2), S(c_2, d_2)\rangle$
  - Insert$\langle R(a_2, b_2, c_3), S(c_3, d_3)\rangle$
  - Insert$\langle R(a_2, b_2, c_2), S(c_2, d_1)\rangle$
  - Insert$\langle R(a_2, b_2, c_0)\rangle$
- The following alternative is not tuple minimal.
  - Insert$\langle R(a_2, b_2, c_2), S(c_2, d_2), R(a_2, b_2, c_3), S(c_3, d_3)\rangle$
- Most existing approaches work only with ground atoms, and so do not provide a formal preference ranking on minimal alternatives.



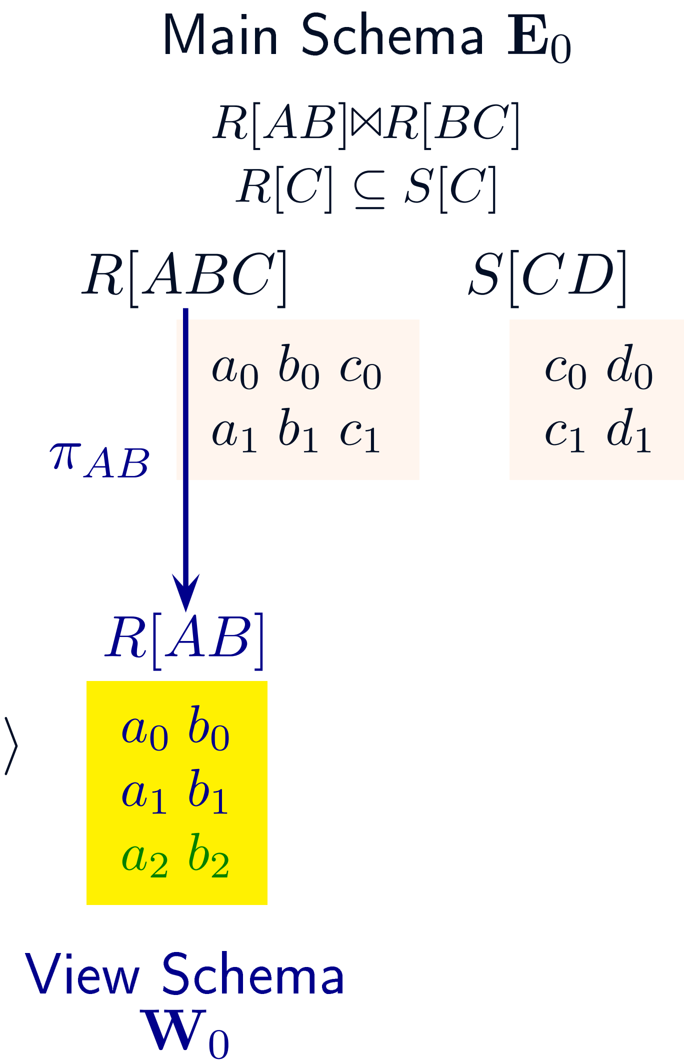
The Idea of Minimal Change

- Consider the update Insert$\langle R(a_2, b_2)\rangle$ into $\mathbf{W}_1$.
- The following alternatives for the reflection are all *tuple minimal* — no proper subset is a solution.
    - Insert$\langle R(a_2, b_2, c_2), S(c_2, d_2)\rangle$
    - Insert$\langle R(a_2, b_2, c_3), S(c_3, d_3)\rangle$
    - Insert$\langle R(a_2, b_2, c_2), S(c_2, d_1)\rangle$
    - Insert$\langle R(a_2, b_2, c_0)\rangle$
- The following alternative is not tuple minimal.
    - Insert$\langle R(a_2, b_2, c_2), S(c_2, d_2), R(a_2, b_2, c_3), S(c_3, d_3)\rangle$
- Most existing approaches work only with ground atoms, and so do not provide a formal preference ranking on minimal alternatives.
- Often, the selection process is left to the user.



The Idea of Minimal Change

- Consider the update Insert$\langle R(a_2, b_2)\rangle$ into $\mathbf{W}_1$.
- The following alternatives for the reflection are all *tuple minimal* — no proper subset is a solution.
  - Insert$\langle R(a_2, b_2, c_2), S(c_2, d_2)\rangle$
  - Insert$\langle R(a_2, b_2, c_3), S(c_3, d_3)\rangle$
  - Insert$\langle R(a_2, b_2, c_2), S(c_2, d_1)\rangle$
  - Insert$\langle R(a_2, b_2, c_0)\rangle$
- The following alternative is not tuple minimal.
  - Insert$\langle R(a_2, b_2, c_2), S(c_2, d_2), R(a_2, b_2, c_3), S(c_3, d_3)\rangle$
- Most existing approaches work only with ground atoms, and so do not provide a formal preference ranking on minimal alternatives.
- Often, the selection process is left to the user.



Question: Is there a reasonable way to measure the quality of tuple-minimal alternatives?

The Information Content of a Database State

Idea: Exploit the first-order properties of the model, rather than just counting the number of tuples which are changed.

Idea: Exploit the first-order properties of the model, rather than just counting the number of tuples which are changed.

- Model database states as finite sets of ground atoms. $\mathrm{DB}(\mathbf{D})$ = set of all database states of schema $\mathbf{D}$.

The Information Content of a Database State

Idea: Exploit the first-order properties of the model, rather than just counting the number of tuples which are changed.

- Model database states as finite sets of ground atoms. $\mathrm{DB}(\mathbf{D})$ = set of all database states of schema $\mathbf{D}$.
- $\mathsf{WFS}(\mathbf{D})$ denotes the set of all sentences in the language of the schema $\mathbf{D}$.

The Information Content of a Database State

Idea: Exploit the first-order properties of the model, rather than just counting the number of tuples which are changed.

- Model database states as finite sets of ground atoms. $\mathsf{DB}(\mathbf{D})$ = set of all database states of schema $\mathbf{D}$.
- $\mathsf{WFS}(\mathbf{D})$ denotes the set of all sentences in the language of the schema $\mathbf{D}$.
- For $\Phi \subseteq \mathsf{WFS}(\mathbf{D})$ and $M \in \mathsf{DB}(\mathbf{D})$, the *information content* of M relative to Φ:

$$\mathsf{Info}\langle M, \Phi \rangle = \{\varphi \in \Phi \mid M \models \varphi\}$$

The Information Content of a Database State

Idea: Exploit the first-order properties of the model, rather than just counting the number of tuples which are changed.

- Model database states as finite sets of ground atoms. $\mathsf{DB}(\mathbf{D})$ = set of all database states of schema $\mathbf{D}$.
- $\mathsf{WFS}(\mathbf{D})$ denotes the set of all sentences in the language of the schema $\mathbf{D}$.
- For $\Phi \subseteq \mathsf{WFS}(\mathbf{D})$ and $M \in \mathsf{DB}(\mathbf{D})$, the *information content* of M relative to Φ:

$$\mathsf{Info}\langle M, \Phi \rangle = \{\varphi \in \Phi \mid M \models \varphi\}$$

- For Φ = ground atoms in $\mathsf{WFS}(\mathbf{D})$, $\mathsf{Info}\langle M, \Phi \rangle = M$.

The Information Content of a Database State

Idea: Exploit the first-order properties of the model, rather than just counting the number of tuples which are changed.

- Model database states as finite sets of ground atoms. $\text{DB}(\mathbf{D})$ = set of all database states of schema $\mathbf{D}$.
- $\text{WFS}(\mathbf{D})$ denotes the set of all sentences in the language of the schema $\mathbf{D}$.
- For $\Phi \subseteq \text{WFS}(\mathbf{D})$ and $M \in \text{DB}(\mathbf{D})$, the *information content* of M relative to Φ:

$$\text{Info}\langle M, \Phi\rangle = \{\varphi \in \Phi \mid M \models \varphi\}$$

- For Φ = ground atoms in $\text{WFS}(\mathbf{D})$, $\text{Info}\langle M, \Phi\rangle = M$.
- For finer measure of information content, a larger subset of $\text{WFS}(\mathbf{D})$ is used.

The Information Content of a Database State

Idea: Exploit the first-order properties of the model, rather than just counting the number of tuples which are changed.

- Model database states as finite sets of ground atoms. $\text{DB}(\mathbf{D})$ = set of all database states of schema $\mathbf{D}$.
- $\text{WFS}(\mathbf{D})$ denotes the set of all sentences in the language of the schema $\mathbf{D}$.
- For $\Phi \subseteq \text{WFS}(\mathbf{D})$ and $M \in \text{DB}(\mathbf{D})$, the *information content* of M relative to Φ:

$$\text{Info}\langle M, \Phi \rangle = \{\varphi \in \Phi \mid M \models \varphi\}$$

- For $\Phi =$ ground atoms in $\text{WFS}(\mathbf{D})$, $\text{Info}\langle M, \Phi \rangle = M$.
- For finer measure of information content, a larger subset of $\text{WFS}(\mathbf{D})$ is used.
- The general idea is to regard optimal reflections as those which minimize the change of information content, rather than just the number of tuples which are changed.

The Information Content of a Database State

Idea: Exploit the first-order properties of the model, rather than just counting the number of tuples which are changed.

- Model database states as finite sets of ground atoms. DB$(\mathbf{D})$ = set of all database states of schema $\mathbf{D}$.
- WFS$(\mathbf{D})$ denotes the set of all sentences in the language of the schema $\mathbf{D}$.
- For $\Phi \subseteq$ WFS$(\mathbf{D})$ and $M \in$ DB$(\mathbf{D})$, the *information content* of M relative to Φ:

$$\text{Info}\langle M, \Phi\rangle = \{\varphi \in \Phi \mid M \models \varphi\}$$

- For $\Phi =$ ground atoms in WFS$(\mathbf{D})$, Info$\langle M, \Phi\rangle = M$.
- For finer measure of information content, a larger subset of WFS$(\mathbf{D})$ is used.
- The general idea is to regard optimal reflections as those which minimize the change of information content, rather than just the number of tuples which are changed.
- To make this concept useful, some further properties are necessary.

Monotonicity of an Information Measure

- In general, inserting a new tuple into a database can result in formulas being removed from the information content.

Monotonicity of an Information Measure

- In general, inserting a new tuple into a database can result in formulas being removed from the information content.
- *Example*: Let $\Phi = \mathsf{WFS}(\mathbf{E}_0)$.

$$
\begin{aligned}
M = \{S(c_0, d_0)\} \quad &\text{implies} \quad (\forall x)(S(x,y) \Rightarrow (x = c_0)) \in \mathsf{Info}\langle M, \Phi\rangle. \\
M' = \{S(c_0, d_0), S(c_1, d_1)\} \quad &\text{implies} \quad (\forall x)(S(x,y) \Rightarrow (x = c_0)) \notin \mathsf{Info}\langle M, \Phi\rangle.
\end{aligned}
$$

Monotonicity of an Information Measure

- In general, inserting a new tuple into a database can result in formulas being removed from the information content.
- *Example*: Let $\Phi = \mathsf{WFS}(\mathbf{E}_0)$.

$$M = \{S(c_0, d_0)\} \quad \text{implies} \quad (\forall x)(S(x,y) \Rightarrow (x = c_0)) \in \mathsf{Info}\langle M, \Phi\rangle.$$
$$M' = \{S(c_0, d_0), S(c_1, d_1)\} \quad \text{implies} \quad (\forall x)(S(x,y) \Rightarrow (x = c_0)) \notin \mathsf{Info}\langle M, \Phi\rangle.$$

- Call Φ *information monotone* if:

$$M_1 \subseteq M_2 \Rightarrow \mathsf{Info}\langle M_1, \Phi\rangle \subseteq \mathsf{Info}\langle M_2, \Phi\rangle$$

Monotonicity of an Information Measure

- In general, inserting a new tuple into a database can result in formulas being removed from the information content.
- *Example*: Let $\Phi = \mathsf{WFS}(\mathbf{E}_0)$.

$$\begin{aligned} M &= \{S(c_0, d_0)\} \quad \text{implies} \quad (\forall x)(S(x,y) \Rightarrow (x = c_0)) \in \mathsf{Info}\langle M, \Phi\rangle. \\ M' &= \{S(c_0, d_0), S(c_1, d_1)\} \quad \text{implies} \quad (\forall x)(S(x,y) \Rightarrow (x = c_0)) \notin \mathsf{Info}\langle M, \Phi\rangle. \end{aligned}$$

- Call Φ *information monotone* if:

$$M_1 \subseteq M_2 \Rightarrow \mathsf{Info}\langle M_1, \Phi\rangle \subseteq \mathsf{Info}\langle M_2, \Phi\rangle$$

- If Φ consists of positive formulas (no negation) and existential (no $\forall$) sentences, then it is automatically information monotone.

Monotonicity of an Information Measure

- In general, inserting a new tuple into a database can result in formulas being removed from the information content.
- *Example*: Let $\Phi = \mathsf{WFS}(\mathbf{E}_0)$.

$$\begin{aligned} M &= \{S(c_0, d_0)\} \quad \text{implies} \quad (\forall x)(S(x,y) \Rightarrow (x = c_0)) \in \mathsf{Info}\langle M, \Phi\rangle. \\ M' &= \{S(c_0, d_0), S(c_1, d_1)\} \quad \text{implies} \quad (\forall x)(S(x,y) \Rightarrow (x = c_0)) \notin \mathsf{Info}\langle M, \Phi\rangle. \end{aligned}$$

- Call Φ *information monotone* if:

$$M_1 \subseteq M_2 \Rightarrow \mathsf{Info}\langle M_1, \Phi\rangle \subseteq \mathsf{Info}\langle M_2, \Phi\rangle$$

- If Φ consists of positive formulas (no negation) and existential (no $\forall$) sentences, then it is automatically information monotone.
- Φ will always be chosen to be information monotone.

Monotonicity of an Information Measure

- In general, inserting a new tuple into a database can result in formulas being removed from the information content.
- *Example*: Let $\Phi = \text{WFS}(\mathbf{E}_0)$.

$$\begin{aligned} M &= \{S(c_0, d_0)\} \quad \text{implies} \quad (\forall x)(S(x,y) \Rightarrow (x = c_0)) \in \text{Info}\langle M, \Phi\rangle. \\ M' &= \{S(c_0, d_0), S(c_1, d_1)\} \quad \text{implies} \quad (\forall x)(S(x,y) \Rightarrow (x = c_0)) \notin \text{Info}\langle M, \Phi\rangle. \end{aligned}$$

- Call Φ *information monotone* if:

$$M_1 \subseteq M_2 \Rightarrow \text{Info}\langle M_1, \Phi\rangle \subseteq \text{Info}\langle M_2, \Phi\rangle$$

- If Φ consists of positive formulas (no negation) and existential (no $\forall$) sentences, then it is automatically information monotone.
- Φ will always be chosen to be information monotone.
- In most cases, it will be chosen to be a subset of $\text{WFS}(\mathbf{D}, \exists\wedge+)$, the set of all existential positive conjunctive sentences in the language of the schema $\mathbf{D}$.

Update Difference and Optimal Reflections

- An *update* is modelled formally as a pair of states

$$(M_1, M_2) = (\text{current state}, \text{next state}).$$

Update Difference and Optimal Reflections

- An *update* is modelled formally as a pair of states
$$(M_1, M_2) = (\text{current state}, \text{next state}).$$
- The *update difference* (w.r.t. Φ) for an update (M_1, M_2) is the change of information associated with that update.

Update Difference and Optimal Reflections

- An *update* is modelled formally as a pair of states
$$(M_1, M_2) = (\text{current state}, \text{next state}).$$
- The *update difference* (w.r.t. Φ) for an update (M_1, M_2) is the change of information associated with that update.
- The *positive*, *negative*, and *total information differences* for (M_1, M_2) w.r.t. Φ are defined as follows:

$$\begin{aligned}
\Delta^{+}\langle (M_1, M_2), \Phi\rangle &= \mathsf{Info}\langle M_2, \Phi\rangle \setminus \mathsf{Info}\langle M_1, \Phi\rangle \\
\Delta^{-}\langle (M_1, M_2), \Phi\rangle &= \mathsf{Info}\langle M_1, \Phi\rangle \setminus \mathsf{Info}\langle M_2, \Phi\rangle \\
\Delta\langle (M_1, M_2), \Phi\rangle &= \Delta^{+}\langle (M_1, M_2), \Phi\rangle \cup \Delta^{-}\langle (M_1, M_2), \Phi\rangle
\end{aligned}$$

Update Difference and Optimal Reflections

- An *update* is modelled formally as a pair of states
$$(M_1, M_2) = (\text{current state}, \text{next state}).$$
- The *update difference* (w.r.t. Φ) for an update (M_1, M_2) is the change of information associated with that update.
- The *positive*, *negative*, and *total information differences* for (M_1, M_2) w.r.t. Φ are defined as follows:

$$\begin{aligned}
\Delta^+\langle(M_1, M_2), \Phi\rangle &= \mathsf{Info}\langle M_2, \Phi\rangle \setminus \mathsf{Info}\langle M_1, \Phi\rangle \\
\Delta^-\langle(M_1, M_2), \Phi\rangle &= \mathsf{Info}\langle M_1, \Phi\rangle \setminus \mathsf{Info}\langle M_2, \Phi\rangle \\
\Delta\langle(M_1, M_2), \Phi\rangle &= \Delta^+\langle(M_1, M_2), \Phi\rangle \cup \Delta^-\langle(M_1, M_2), \Phi\rangle
\end{aligned}$$

- Observe that if $\Phi = \mathsf{WFS}(\mathbf{D}, \mathsf{Atoms})$, then the update difference reduces to the set of changes (tuples inserted or deleted) by the update.

Update Difference and Optimal Reflections

- An *update* is modelled formally as a pair of states
$$(M_1, M_2) = (\text{current state}, \text{next state}).$$
- The *update difference* (w.r.t. Φ) for an update (M_1, M_2) is the change of information associated with that update.
- The *positive*, *negative*, and *total information differences* for (M_1, M_2) w.r.t. Φ are defined as follows:

$$\begin{aligned}
\Delta^{+}\langle (M_1, M_2), \Phi \rangle &= \mathsf{Info}\langle M_2, \Phi \rangle \setminus \mathsf{Info}\langle M_1, \Phi \rangle \\
\Delta^{-}\langle (M_1, M_2), \Phi \rangle &= \mathsf{Info}\langle M_1, \Phi \rangle \setminus \mathsf{Info}\langle M_2, \Phi \rangle \\
\Delta\langle (M_1, M_2), \Phi \rangle &= \Delta^{+}\langle (M_1, M_2), \Phi \rangle \cup \Delta^{-}\langle (M_1, M_2), \Phi \rangle
\end{aligned}$$

- Observe that if $\Phi = \mathsf{WFS}(\mathbf{D}, \mathsf{Atoms})$, then the update difference reduces to the set of changes (tuples inserted or deleted) by the update.
- An *optimal reflection* of a view update is a tuple-minimal reflection to the main schema for which the update difference is least.

The Choice of Information Measure

- The key idea is to render Φ indifferent to the names of new constants which are inserted.

The Choice of Information Measure

- The key idea is to render Φ indifferent to the names of new constants which are inserted.
- *Setting*: Main schema $= \mathbf{D}$, View $= (\mathbf{V}, \gamma : \mathbf{D} \to \mathbf{V})$.
- Let:
  - $M_1 =$ the initial state of the main schema.
  - $(\gamma(M_1), N_2)$ the desired update to the view.

The Choice of Information Measure

- The key idea is to render Φ indifferent to the names of new constants which are inserted.
- *Setting*: Main schema $= \mathbf{D}$, View $= (\mathbf{V}, \gamma : \mathbf{D} \to \mathbf{V})$.
- Let:
  - $M_1 =$ the initial state of the main schema.
  - $(\gamma(M_1), N_2)$ the desired update to the view.
- Define ConstSym$(M_1 \cup \gamma(M_1) \cup N_2)$ to be the set of all constant symbols which occur in these databases.

The Choice of Information Measure

- The key idea is to render Φ indifferent to the names of new constants which are inserted.
- *Setting*: Main schema $= \mathbf{D}$, View $= (\mathbf{V}, \gamma : \mathbf{D} \to \mathbf{V})$.
- Let:
  - $M_1 =$ the initial state of the main schema.
  - $(\gamma(M_1), N_2)$ the desired update to the view.
- Define $\mathsf{ConstSym}(M_1 \cup \gamma(M_1) \cup N_2)$ to be the set of all constant symbols which occur in these databases.
- For the information measure, choose:

$$\Phi = \mathsf{WFS}(\mathbf{D}, \exists\wedge+, \mathsf{ConstSym}(M_1 \cup \gamma(M_1) \cup N_2)),$$

  the positive conjunctive sentences in the language of the main schema $\mathbf{D}$ which involve only those constant symbols which occur in at least one of the three databases.

The Choice of Information Measure

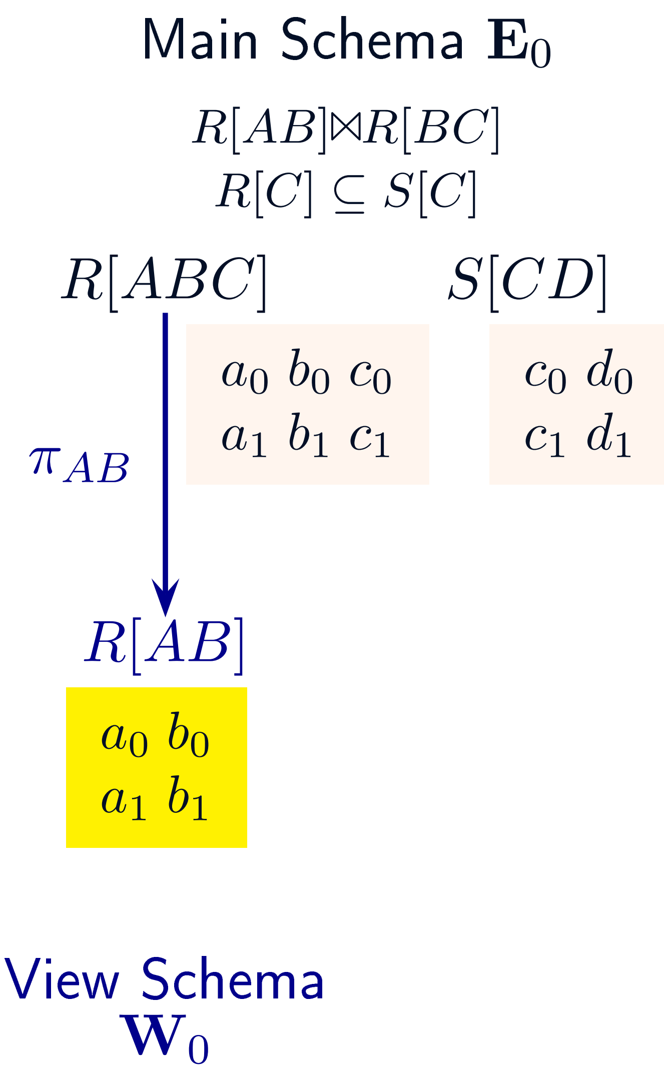
- The key idea is to render Φ indifferent to the names of new constants which are inserted.
- *Setting*: Main schema $= \mathbf{D}$, View $= (\mathbf{V}, \gamma : \mathbf{D} \to \mathbf{V})$.
- Let:
  - $M_1 =$ the initial state of the main schema.
  - $(\gamma(M_1), N_2)$ the desired update to the view.
- Define ConstSym$(M_1 \cup \gamma(M_1) \cup N_2)$ to be the set of all constant symbols which occur in these databases.
- For the information measure, choose:

$$\Phi = \text{WFS}(\mathbf{D}, \exists\wedge+, \text{ConstSym}(M_1 \cup \gamma(M_1) \cup N_2)),$$

  the positive conjunctive sentences in the language of the main schema $\mathbf{D}$ which involve only those constant symbols which occur in at least one of the three databases.
- Such formulas are indifferent to the identities of new constants which are inserted.

- In the example to the left, if the initial state of $\mathbf{E}_0$ is denoted M_{00} , then:

$$\mathrm{ConstSym}(M_{00}) = \{a_0, a_1, b_0, b_1, c_0, c_1, d_0, d_1\}$$



Examples of Measures for Information Content

- In the example to the left, if the initial state of $\mathbf{E}_0$ is denoted M_{00}, then:

$$\mathsf{ConstSym}(M_{00}) = \{a_0, a_1, b_0, b_1, c_0, c_1, d_0, d_1\}$$

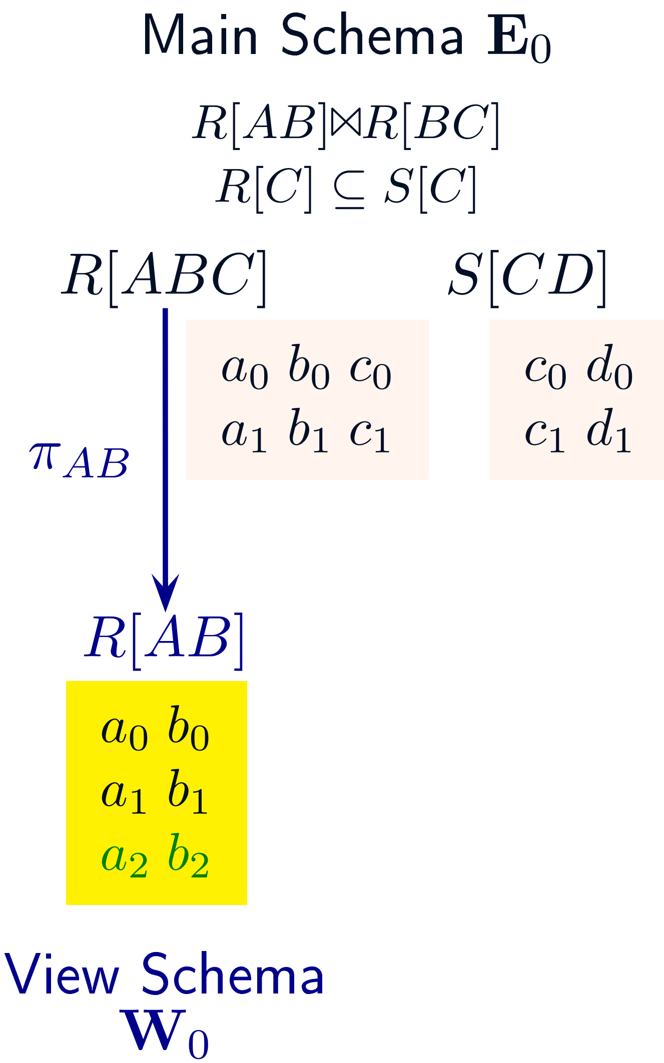
- For the view update

$$\mathsf{Insert}\langle\{R(a_2, b_2)\}\rangle =$$

$$(\{(a_0, b_0), (a_1, b_1)\}, \{(a_0, b_0), (a_1, b_1), (a_2, b_2)\}$$

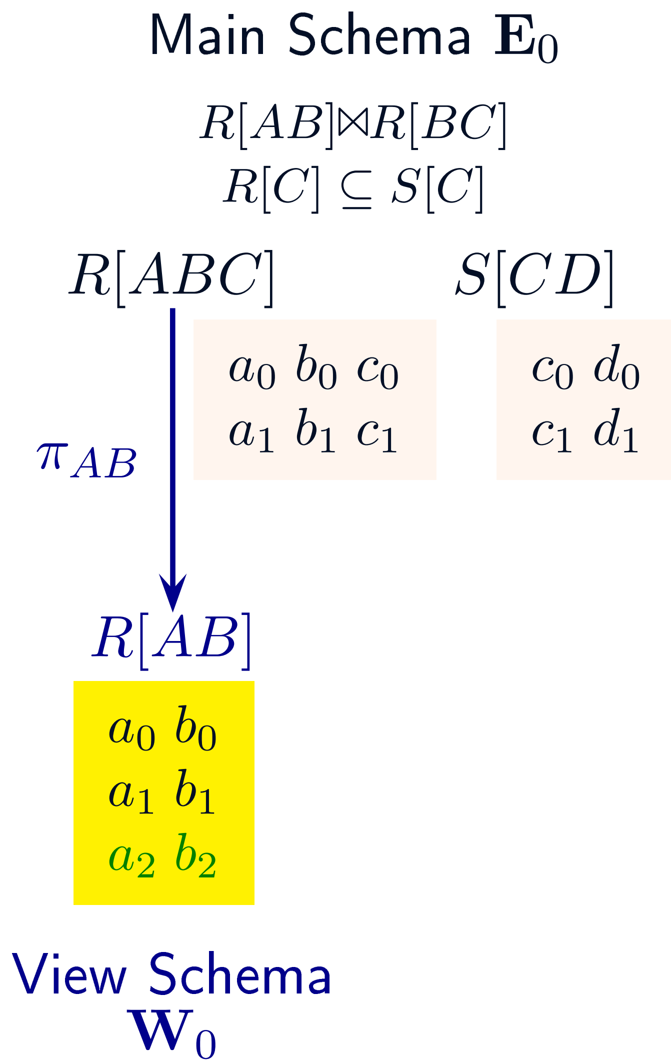
the set of constants which are allowed in the sentences defining the information of the new state of $\mathbf{E}_0$ is:

$$\mathsf{ConstSym}(M_{00}) \cup \{a_2, b_2\}$$



Examples of Information Measure

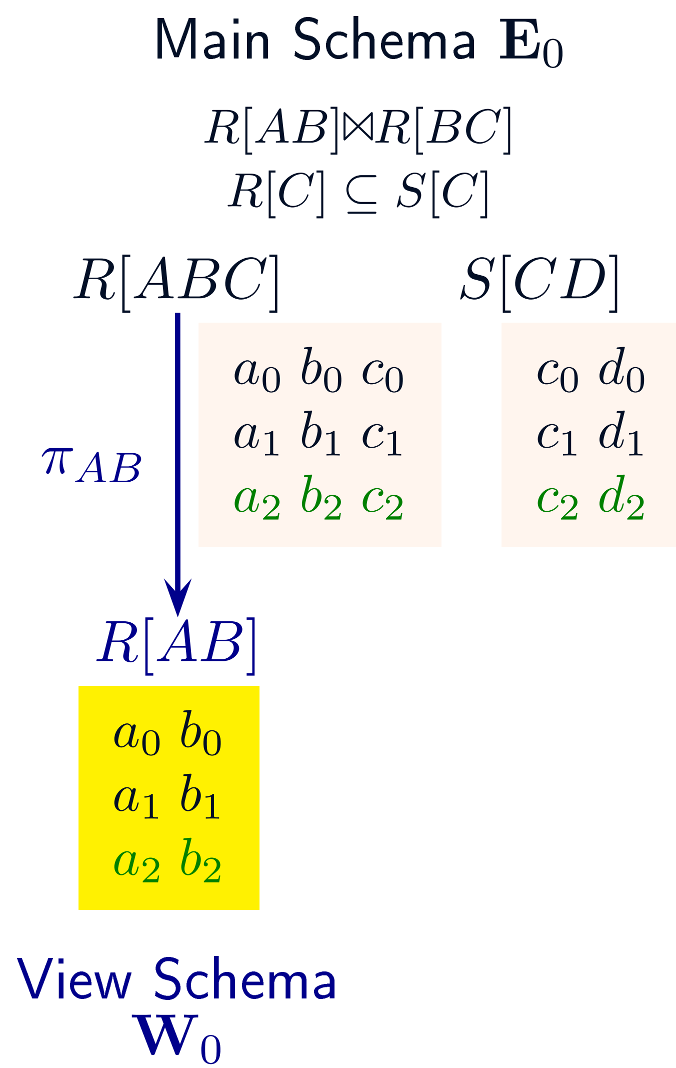
- Consider again the view update Insert$\langle R(a_2, b_2)\rangle$.



Examples of Information Measure

- Consider again the view update Insert$\langle R(a_2, b_2)\rangle$.
- Consider the reflection

  Insert$\langle R(a_2, b_2, c_2), S(c_2, d_2)\rangle$ to $\mathbf{E}_0$.

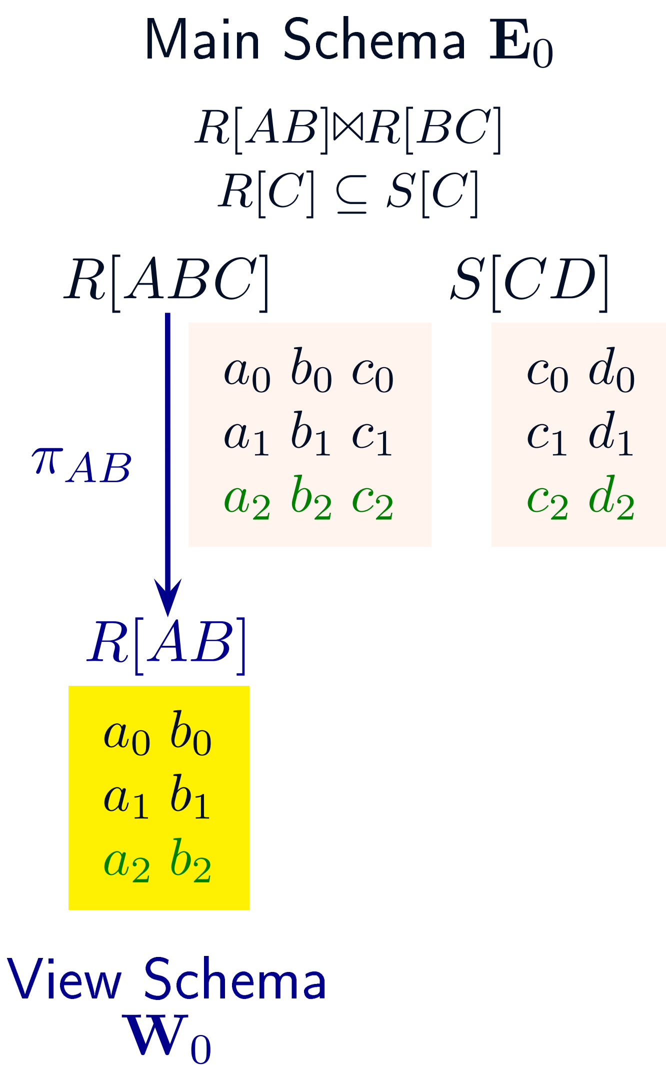


Examples of Information Measure

- Consider again the view update Insert$\langle R(a_2, b_2)\rangle$.
- Consider the reflection

  Insert$\langle R(a_2, b_2, c_2), S(c_2, d_2)\rangle$ to $\mathbf{E}_0$.
- A *basis* for the information content is

$$M_{00} \cup \{(\exists x)(\exists y)(R(a_2, b_2, x) \wedge S(x, y))\}$$



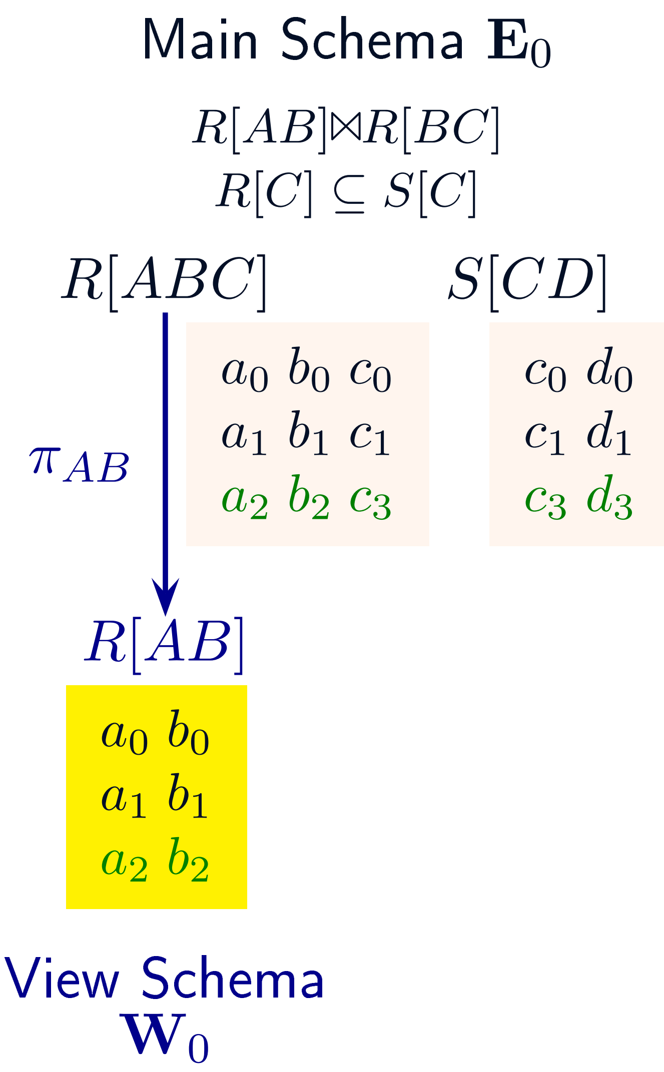
Examples of Information Measure

- Consider again the view update Insert$\langle R(a_2, b_2)\rangle$.
- Consider the reflection

  Insert$\langle R(a_2, b_2, c_2), S(c_2, d_2)\rangle$ to $\mathbf{E}_0$.
- A *basis* for the information content is

$$M_{00} \cup \{(\exists x)(\exists y)(R(a_2, b_2, x) \wedge S(x, y))\}$$

- The reflection Insert$\langle R(a_2, b_2, c_3), S(c_3, d_3)\rangle$ has the same basis.



Examples of Information Measure

- Consider again the view update Insert$\langle R(a_2, b_2)\rangle$.
- Consider the reflection

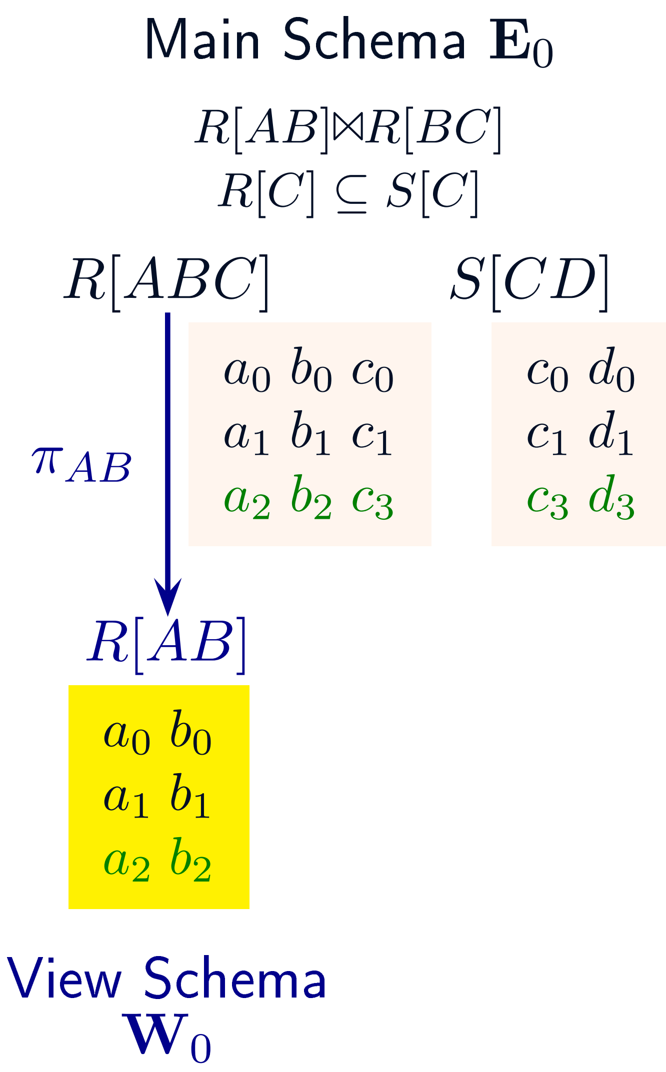
  Insert$\langle R(a_2, b_2, c_2), S(c_2, d_2)\rangle$ to $\mathbf{E}_0$.
- A *basis* for the information content is

$$M_{00} \cup \{(\exists x)(\exists y)(R(a_2, b_2, x) \wedge S(x, y))\}$$

- The reflection Insert$\langle R(a_2, b_2, c_3), S(c_3, d_3)\rangle$ has the same basis.
- These two reflections are *equivalent* with respect to

  WFS$(\mathbf{E}_0, \exists\wedge+, \{a_0, a_1, a_2, b_0, b_1, b_2, c_0, c_1, d_0, d_1\})$,

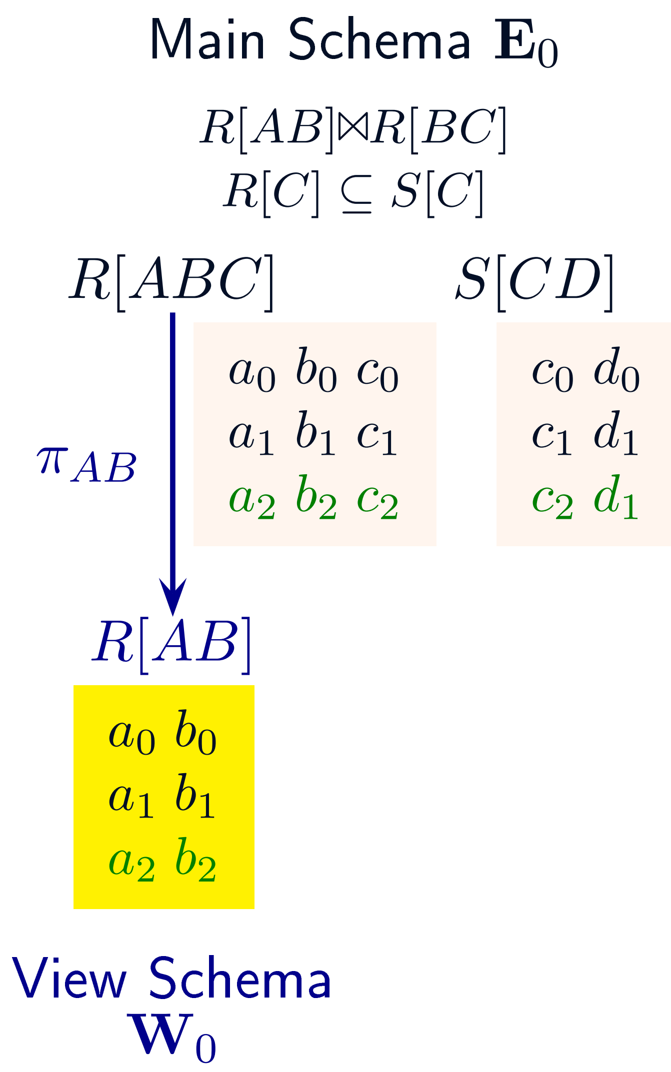
  but not with respect to WFS$(\mathbf{E}_0, \exists\wedge+)$.



Examples of Information Measure – Part 2

- Now consider the reflection

$$\text{Insert}\langle R(a_2, b_2, c_2), S(c_2, d_1)\rangle \text{ to } \mathbf{E}_0.$$

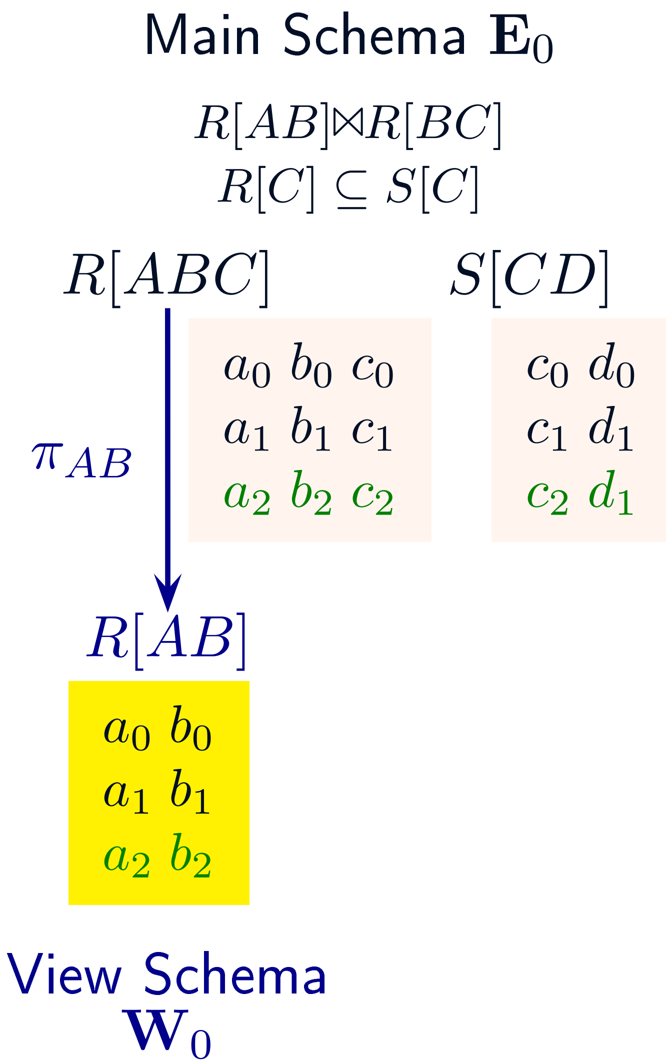


- Now consider the reflection

  $$\text{Insert}\langle R(a_2, b_2, c_2), S(c_2, d_1)\rangle \text{ to } \mathbf{E}_0.$$

- A basis for the information content is

$$M_{00} \cup \{(\exists x)(R(a_2, b_2, x) \wedge S(x, d_1))\}$$



- Now consider the reflection

$$\text{Insert}\langle R(a_2, b_2, c_2), S(c_2, d_1)\rangle \text{ to } \mathbf{E}_0.$$

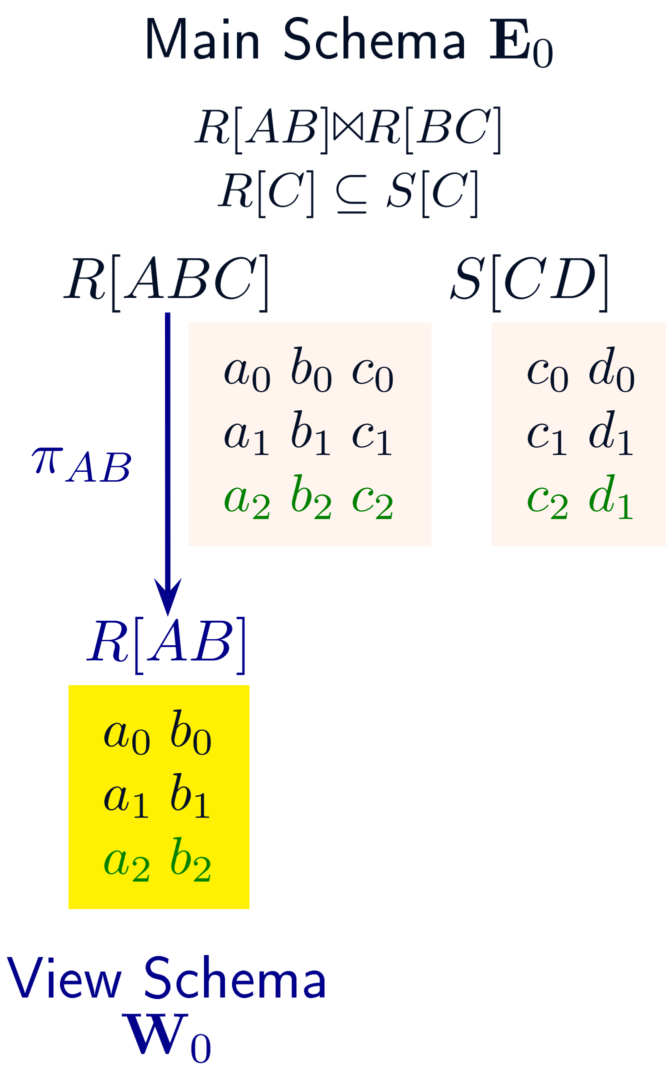
- A basis for the information content is

$$M_{00} \cup \{(\exists x)(R(a_2, b_2, x) \wedge S(x, d_1))\}$$

- This reflection is not optimal, since

$$M_{00} \cup \{(\exists x)(R(a_2, b_2, x) \wedge S(x, d_3))\}$$
$$\models M_{00} \cup \{(\exists x)(\exists y)(R(a_2, b_2, x) \wedge S(x, y))\}$$

but not conversely.



- Now consider the reflection

$$\text{Insert}\langle R(a_2, b_2, c_2), S(c_2, d_1)\rangle \text{ to } \mathbf{E}_0.$$

- A basis for the information content is

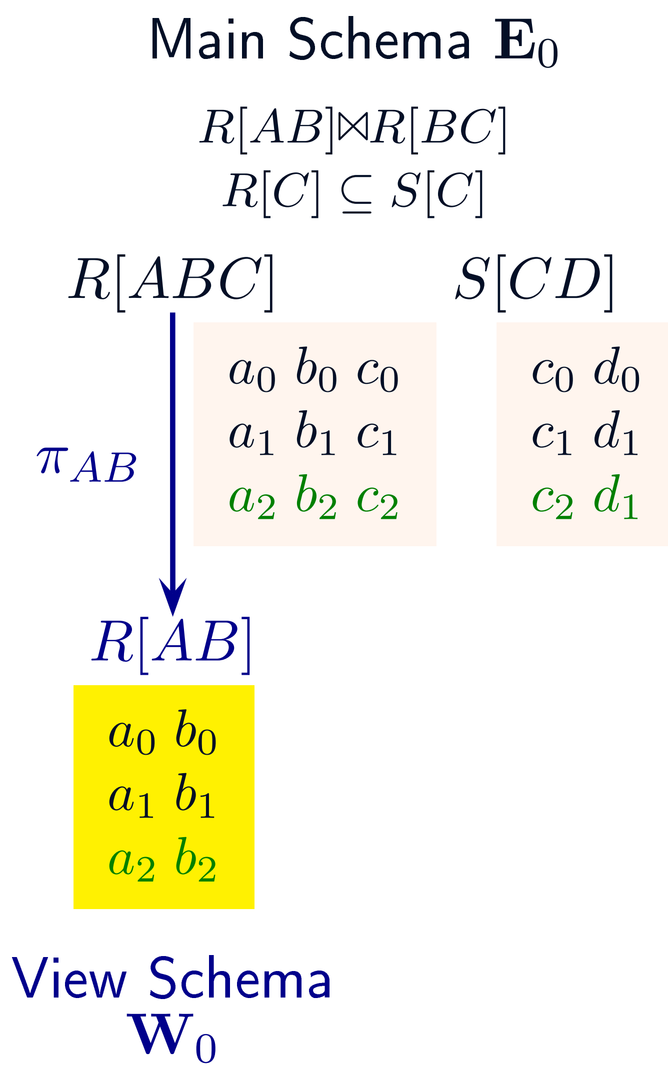
$$M_{00} \cup \{(\exists x)(R(a_2, b_2, x) \wedge S(x, d_1))\}$$

- This reflection is not optimal, since

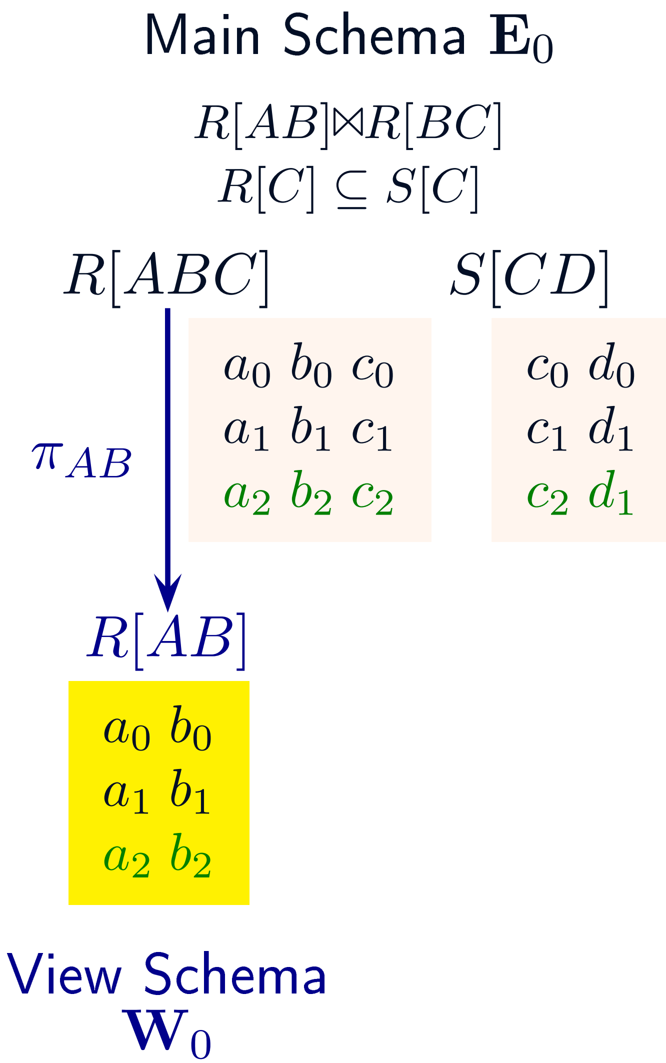
$$M_{00} \cup \{(\exists x)(R(a_2, b_2, x) \wedge S(x, d_3))\}$$
$$\models M_{00} \cup \{(\exists x)(\exists y)(R(a_2, b_2, x) \wedge S(x, y))\}$$

  but not conversely.

- Inserting $S(c_2, d_1)$ adds strictly more information than inserting $S(c_2, d_2)$.

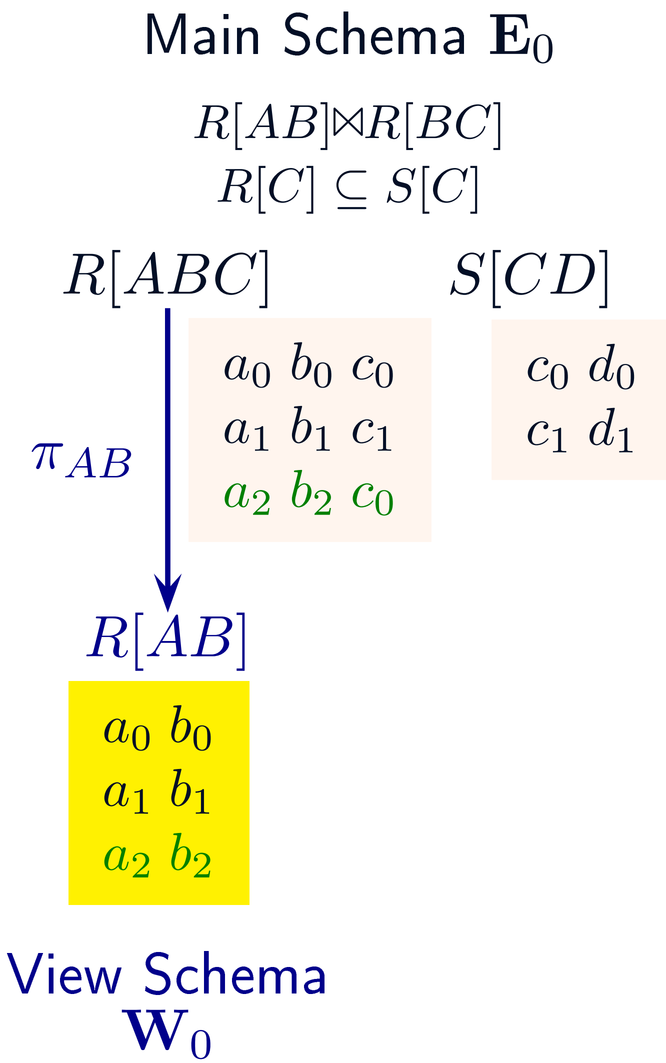


- Now consider the reflection
  $$\text{Insert}\langle R(a_2, b_2, c_2), S(c_2, d_1)\rangle \text{ to } \mathbf{E}_0.$$
- A basis for the information content is
  $$M_{00} \cup \{(\exists x)(R(a_2, b_2, x) \wedge S(x, d_1))\}$$
- This reflection is not optimal, since
  $$M_{00} \cup \{(\exists x)(R(a_2, b_2, x) \wedge S(x, d_3))\}$$
  $$\models M_{00} \cup \{(\exists x)(\exists y)(R(a_2, b_2, x) \wedge S(x, y))\}$$
  but not conversely.
- Inserting $S(c_2, d_1)$ adds strictly more information than inserting $S(c_2, d_2)$.
- Note that this distinction is not possible with simple minimization of the number of atoms which are changed.



- Finally, consider the reflection

$$\text{Insert}\langle R(a_2, b_2, c_0), S(c_0, d_0)\rangle \text{ to } \mathbf{E}_0.$$

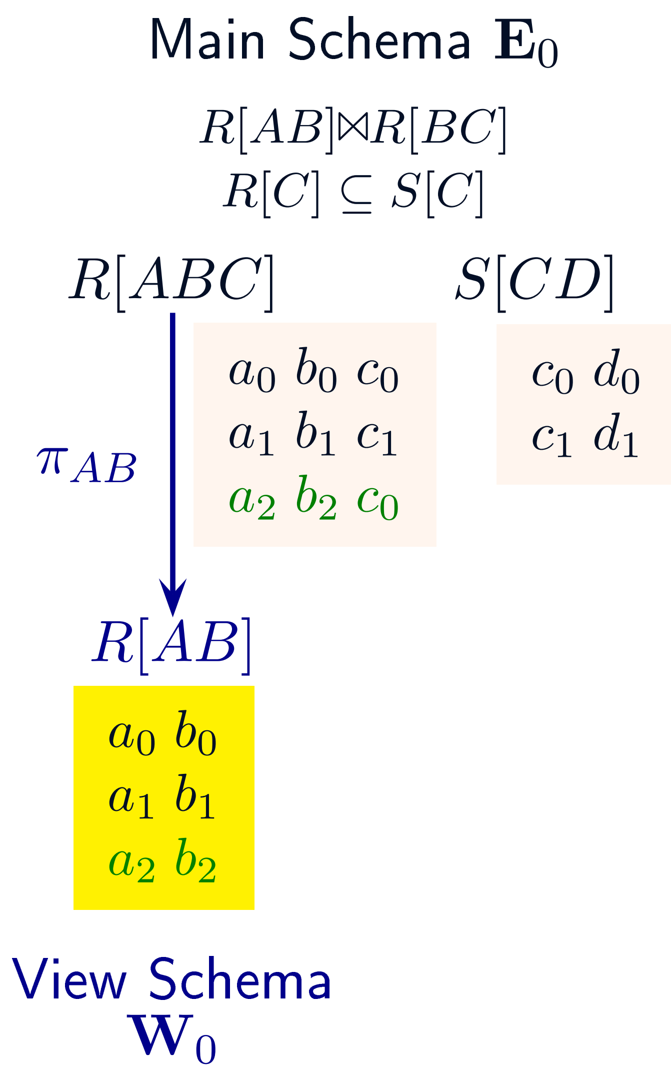


- Finally, consider the reflection

$$\text{Insert}\langle R(a_2, b_2, c_0), S(c_0, d_0)\rangle \text{ to } \mathbf{E}_0.$$

- A basis for the information content is

$$M_{00} \cup \{R(a_2, b_2, c_0)\}$$



Examples of Information Measure – Part 3

- Finally, consider the reflection

$$\text{Insert}\langle R(a_2, b_2, c_0), S(c_0, d_0)\rangle \text{ to } \mathbf{E}_0.$$

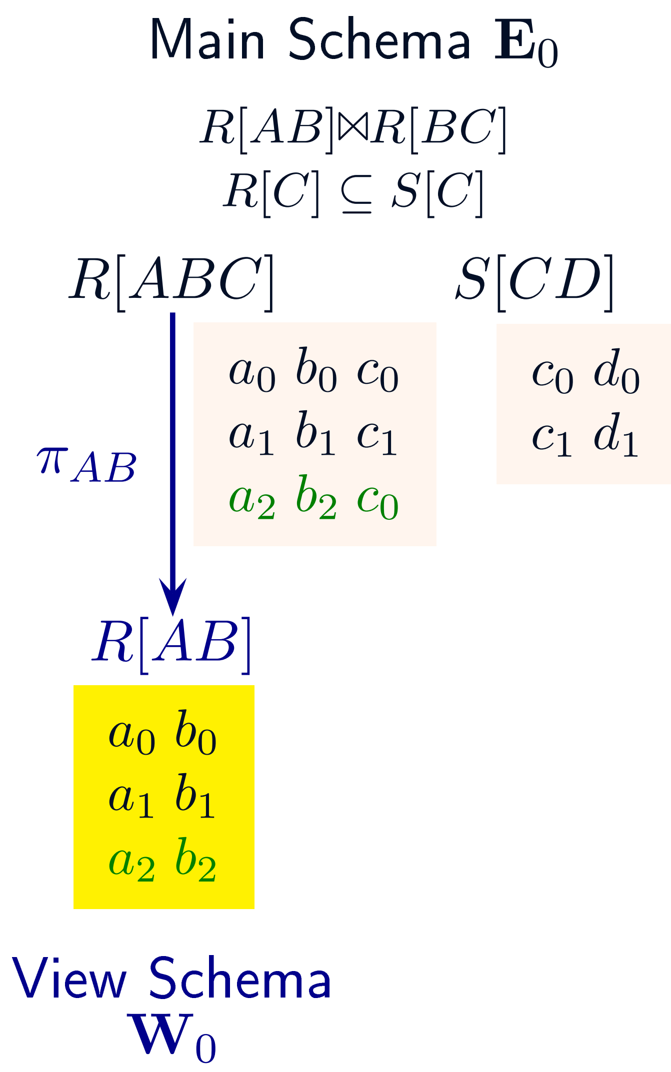
- A basis for the information content is

$$M_{00} \cup \{R(a_2, b_2, c_0)\}$$

- This reflection is not optimal, since

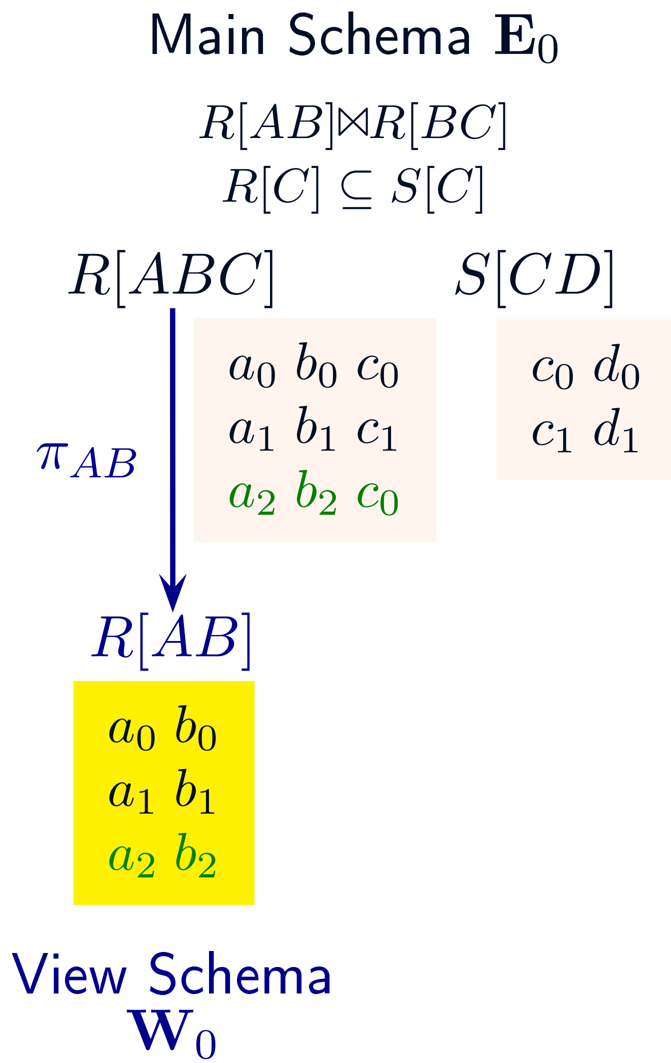
$$M_{00} \cup \{R(a_2, b_2, c_0)\} \models M_{00} \cup \{(\exists x)(\exists y)(R(a_2, b_2, x) \wedge S(x, y))\}$$

but not conversely.



Examples of Information Measure – Part 3

- Finally, consider the reflection
$$\text{Insert}\langle R(a_2, b_2, c_0), S(c_0, d_0)\rangle \text{ to } \mathbf{E}_0.$$
- A basis for the information content is
$$M_{00} \cup \{R(a_2, b_2, c_0)\}$$
- This reflection is not optimal, since
$$M_{00} \cup \{R(a_2, b_2, c_0)\} \models M_{00} \cup \{(\exists x)(\exists y)(R(a_2, b_2, x) \wedge S(x, y))\}$$
but not conversely.
- Note that this choice is suboptimal with respect to information measure even though it inserts fewer tuples than the optimal solution.

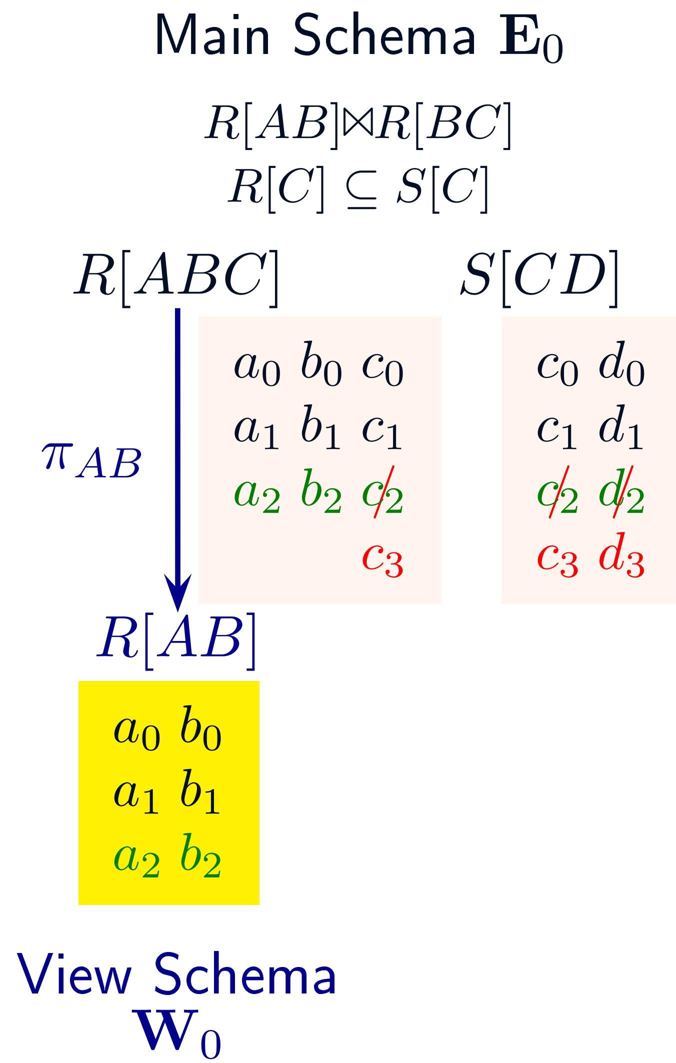


- Note that all of the other reflections may be realized as *endomorphic images* of the first.

$$\text{Insert}\langle R(a_2, b_2, c_2), S(c_2, d_2)\rangle \overset{c_3/c_2,\, d_3/d_2}{\longmapsto} \text{Insert}\langle R(a_2, b_2, c_3), S(c_2, d_3)\rangle$$

$$\text{Insert}\langle R(a_2, b_2, c_2), S(c_2, d_2)\rangle \overset{d_0/d_2}{\longmapsto} \text{Insert}\langle R(a_2, b_2, c_2), S(c_2, d_0)\rangle$$

$$\text{Insert}\langle R(a_2, b_2, c_2), S(c_2, d_2)\rangle \overset{c_0/c_2,\, d_0/d_2}{\longmapsto} \text{Insert}\langle R(a_2, b_2, c_0)\rangle$$



Endomorphisms of Constants and the Associated DB Mapping

- Note that all of the other reflections may be realized as *endomorphic images* of the first.

Insert$\langle R(a_2, b_2, c_2), S(c_2, d_2)\rangle$

$\overset{c_3/c_2,\, d_3/d_2}{\longmapsto}$ Insert$\langle R(a_2, b_2, c_3), S(c_2, d_3)\rangle$

Insert$\langle R(a_2, b_2, c_2), S(c_2, d_2)\rangle$

$\overset{d_0/d_2}{\longmapsto}$ Insert$\langle R(a_2, b_2, c_2), S(c_2, d_0)\rangle$

Insert$\langle R(a_2, b_2, c_2), S(c_2, d_2)\rangle$

$\overset{c_0/c_2,\, d_0/d_2}{\longmapsto}$ Insert$\langle R(a_2, b_2, c_0)\rangle$

Main Schema $\mathbf{E}_0$

$R[AB] \bowtie R[BC]$

$R[C] \subseteq S[C]$

$R[ABC]$

a_0	b_0	c_0
a_1	b_1	c_1
a_2	b_2	c_2

$S[CD]$

c_0	d_0
c_1	d_1
c_2	~~d_2~~ d_0

π_{AB} ↓

$R[AB]$

a_0	b_0
a_1	b_1
a_2	b_2

View Schema $\mathbf{W}_0$

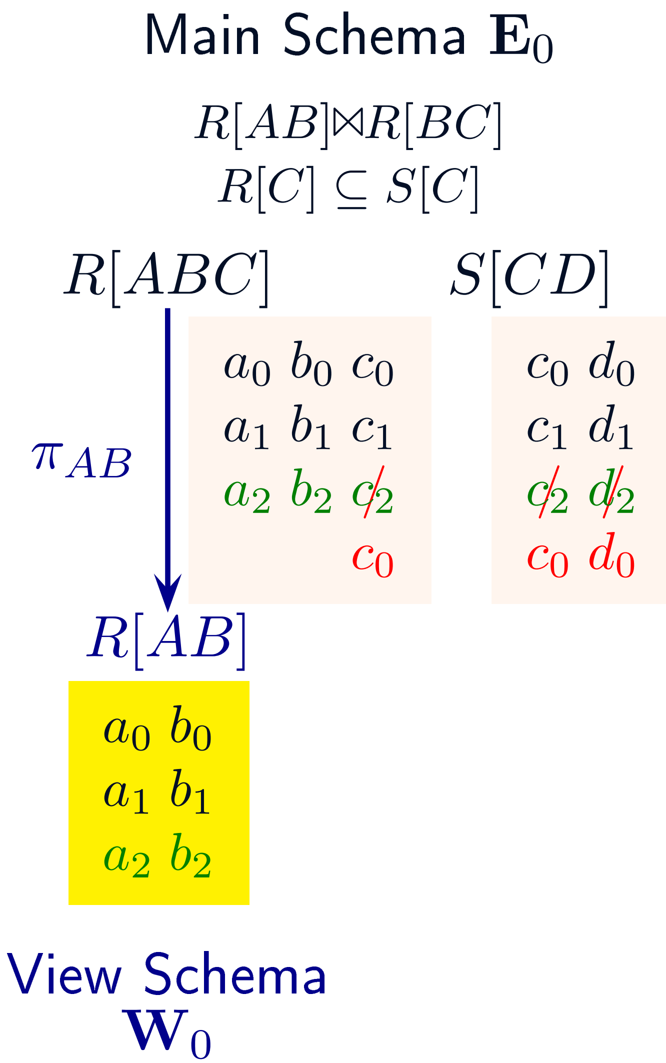
Endomorphisms of Constants and the Associated DB Mapping

- Note that all of the other reflections may be realized as *endomorphic images* of the first.

$$\text{Insert}\langle R(a_2, b_2, c_2), S(c_2, d_2)\rangle \overset{c_3/c_2,\, d_3/d_2}{\longmapsto} \text{Insert}\langle R(a_2, b_2, c_3), S(c_2, d_3)\rangle$$

$$\text{Insert}\langle R(a_2, b_2, c_2), S(c_2, d_2)\rangle \overset{d_0/d_2}{\longmapsto} \text{Insert}\langle R(a_2, b_2, c_2), S(c_2, d_0)\rangle$$

$$\text{Insert}\langle R(a_2, b_2, c_2), S(c_2, d_2)\rangle \overset{c_0/c_2,\, d_0/d_2}{\longmapsto} \text{Insert}\langle R(a_2, b_2, c_0)\rangle$$



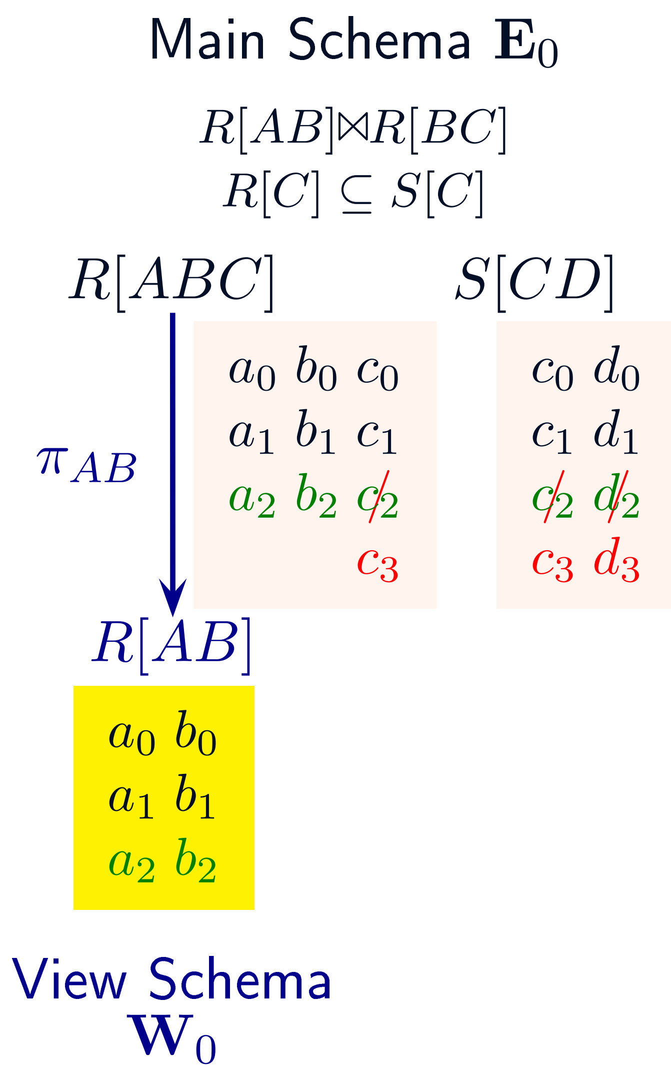
- Note that all of the other reflections may be realized as *endomorphic images* of the first.

$$\text{Insert}\langle R(a_2,b_2,c_2), S(c_2,d_2)\rangle \overset{c_3/c_2,\, d_3/d_2}{\longmapsto} \text{Insert}\langle R(a_2,b_2,c_3), S(c_2,d_3)\rangle$$

$$\text{Insert}\langle R(a_2,b_2,c_2), S(c_2,d_2)\rangle \overset{d_0/d_2}{\longmapsto} \text{Insert}\langle R(a_2,b_2,c_2), S(c_2,d_0)\rangle$$

$$\text{Insert}\langle R(a_2,b_2,c_2), S(c_2,d_2)\rangle \overset{c_0/c_2,\, d_0/d_2}{\longmapsto} \text{Insert}\langle R(a_2,b_2,c_0)\rangle$$

- Note also that the first endomorphism may be reversed, but the others may not.



- There is a natural algebraic structure on the collection of reflections of a given view update.

Endomorphisms of Constants and the Associated DB Mapping

- There is a natural algebraic structure on the collection of reflections of a given view update.
- Let Const$(\mathcal{D})$ denote the set of all constants which can occur in the main schema and the views.

Endomorphisms of Constants and the Associated DB Mapping

- There is a natural algebraic structure on the collection of reflections of a given view update.
- Let Const$(\mathcal{D})$ denote the set of all constants which can occur in the main schema and the views.
- A *constant endomorphism* is a function $h : \text{Const}(\mathcal{D}) \to \text{Const}(\mathcal{D})$ (subject to certain typing constraints which will not be elaborated here).
    - Such mappings are also called *homomorphisms*.

Endomorphisms of Constants and the Associated DB Mapping

- There is a natural algebraic structure on the collection of reflections of a given view update.
- Let $\text{Const}(\mathcal{D})$ denote the set of all constants which can occur in the main schema and the views.
- A *constant endomorphism* is a function $h : \text{Const}(\mathcal{D}) \to \text{Const}(\mathcal{D})$ (subject to certain typing constraints which will not be elaborated here).
  - Such mappings are also called *homomorphisms*.
- Such an endomorphism induces a mapping of tuples via

$$(a_1, a_2, \ldots, a_n) \mapsto (h(a_1), h(a_2), \ldots, h(a_n)).$$

Endomorphisms of Constants and the Associated DB Mapping

- There is a natural algebraic structure on the collection of reflections of a given view update.
- Let Const$(\mathcal{D})$ denote the set of all constants which can occur in the main schema and the views.
- A *constant endomorphism* is a function $h : \text{Const}(\mathcal{D}) \to \text{Const}(\mathcal{D})$ (subject to certain typing constraints which will not be elaborated here).
    - Such mappings are also called *homomorphisms*.
- Such an endomorphism induces a mapping of tuples via $(a_1, a_2, \ldots, a_n) \mapsto (h(a_1), h(a_2), \ldots, h(a_n))$.
- This, in turn, induces a mapping of databases via $M \mapsto \{h(t) \mid t \in M\}$.

Endomorphisms of Constants and the Associated DB Mapping

- There is a natural algebraic structure on the collection of reflections of a given view update.
- Let $\text{Const}(\mathcal{D})$ denote the set of all constants which can occur in the main schema and the views.
- A *constant endomorphism* is a function $h : \text{Const}(\mathcal{D}) \to \text{Const}(\mathcal{D})$ (subject to certain typing constraints which will not be elaborated here).
  - Such mappings are also called *homomorphisms*.
- Such an endomorphism induces a mapping of tuples via
$$(a_1, a_2, \ldots, a_n) \mapsto (h(a_1), h(a_2), \ldots, h(a_n)).$$
- This, in turn, induces a mapping of databases via $M \mapsto \{h(t) \mid t \in M\}$.
- For $A \subseteq \text{Const}(\mathcal{D})$, call h *A-invariant* if $h(a) = a$ for all $a \in A$.

Endomorphisms of Constants and the Associated DB Mapping

- There is a natural algebraic structure on the collection of reflections of a given view update.
- Let $\mathrm{Const}(\mathcal{D})$ denote the set of all constants which can occur in the main schema and the views.
- A *constant endomorphism* is a function $h : \mathrm{Const}(\mathcal{D}) \to \mathrm{Const}(\mathcal{D})$ (subject to certain typing constraints which will not be elaborated here).
  - Such mappings are also called *homomorphisms*.
- Such an endomorphism induces a mapping of tuples via $(a_1, a_2, \ldots, a_n) \mapsto (h(a_1), h(a_2), \ldots, h(a_n))$.
- This, in turn, induces a mapping of databases via $M \mapsto \{h(t) \mid t \in M\}$.
- For $A \subseteq \mathrm{Const}(\mathcal{D})$, call h *A-invariant* if $h(a) = a$ for all $a \in A$.
- For $A \subseteq \mathrm{Const}(\mathcal{D})$, call h *at most A-variant* if it is $(\mathrm{Const}(\mathcal{D}) \setminus A)$-invariant.

Algebraic Representation of Optimal Insertions

Context: $\mathbf{D}$ = relational schema $(\mathbf{V}, \gamma : \mathbf{D} \rightarrow \mathbf{V})$ = a view of $\mathbf{D}$

$M_1 \in \mathsf{DB}(\mathbf{D})$ $(\gamma(M_1), N_2)$ = an insertion on $\mathbf{V}$.

Algebraic Representation of Optimal Insertions

Context: $\mathbf{D}$ = relational schema $(\mathbf{V}, \gamma : \mathbf{D} \to \mathbf{V})$ = a view of $\mathbf{D}$

$M_1 \in \mathsf{DB}(\mathbf{D})$ $(\gamma(M_1), N_2)$ = an insertion on $\mathbf{V}$.

Theorem: Let $M_2 \in \mathsf{DB}(\mathbf{D})$. Then (M_1, M_2) is an optimal reflection of $(\gamma(N_1), N_2)$ iff for every minimal reflection $(\gamma(N_1), M_2')$, there is a unique invariant endomorphism h and with $h(M_2) = M_2'$ and which is at most $(\mathsf{ConstSym}(M_2') \setminus \mathsf{ConstSym}(M_2))$-variant. $\square$

Context: $\mathbf{D}$ = relational schema $(\mathbf{V}, \gamma : \mathbf{D} \to \mathbf{V})$ = a view of $\mathbf{D}$

$M_1 \in \mathsf{DB}(\mathbf{D})$ $(\gamma(M_1), N_2)$ = an insertion on $\mathbf{V}$.

Theorem: Let $M_2 \in \mathsf{DB}(\mathbf{D})$. Then (M_1, M_2) is an optimal reflection of $(\gamma(N_1), N_2)$ iff for every minimal reflection $(\gamma(N_1), M_2')$, there is a unique invariant endomorphism h and with $h(M_2) = M_2'$ and which is at most $(\mathsf{ConstSym}(M_2') \setminus \mathsf{ConstSym}(M_2))$-variant. $\square$

- An optimal reflection is *initial* amongst all minimal reflections.

Context: $\mathbf{D}$ = relational schema $(\mathbf{V}, \gamma : \mathbf{D} \to \mathbf{V})$ = a view of $\mathbf{D}$

$M_1 \in \mathsf{DB}(\mathbf{D})$ $(\gamma(M_1), N_2)$ = an insertion on $\mathbf{V}$.

Theorem: Let $M_2 \in \mathsf{DB}(\mathbf{D})$. Then (M_1, M_2) is an optimal reflection of $(\gamma(N_1), N_2)$ iff for every minimal reflection $(\gamma(N_1), M_2')$, there is a unique invariant endomorphism h and with $h(M_2) = M_2'$ and which is at most $(\mathsf{ConstSym}(M_2') \setminus \mathsf{ConstSym}(M_2))$-variant. □

- An optimal reflection is *initial* amongst all minimal reflections.

Corollary Any two optimal insertions are isomorphic up to renaming of the newly introduced constant symbols. □

Existence of Optimal Insertions

- *Question*: Under what conditions are optimal insertions guaranteed to exist?

Existence of Optimal Insertions

- *Question*: Under what conditions are optimal insertions guaranteed to exist?
- An XEID (extended embedded implicational dependency) [Fagin82 JACM] is one of the following form:

$$(\forall x_1)(\forall x_2)\dots(\forall x_n)((A_1 \wedge A_2 \wedge \dots \wedge A_n) \Rightarrow (\exists y_1)(\exists y_2)\dots(\exists y_r)(B_1 \wedge B_2 \wedge \dots \wedge B_s))$$

Existence of Optimal Insertions

- *Question*: Under what conditions are optimal insertions guaranteed to exist?
- An XEID (extended embedded implicational dependency) [Fagin82 JACM] is one of the following form:

$$(\forall x_1)(\forall x_2)\ldots(\forall x_n)((A_1 \wedge A_2 \wedge \ldots \wedge A_n) \Rightarrow (\exists y_1)(\exists y_2)\ldots(\exists y_r)(B_1 \wedge B_2 \wedge \ldots \wedge B_s))$$

  - Each A_i is a relational atom.

Existence of Optimal Insertions

- *Question*: Under what conditions are optimal insertions guaranteed to exist?
- An XEID (extended embedded implicational dependency) [Fagin82 JACM] is one of the following form:

$$(\forall x_1)(\forall x_2)\ldots(\forall x_n)((A_1 \wedge A_2 \wedge \ldots \wedge A_n) \Rightarrow (\exists y_1)(\exists y_2)\ldots(\exists y_r)(B_1 \wedge B_2 \wedge \ldots \wedge B_s))$$

  - Each A_i is a relational atom.
  - Each B_i is a relational atom or an equality.

Existence of Optimal Insertions

- *Question*: Under what conditions are optimal insertions guaranteed to exist?
- An XEID (extended embedded implicational dependency) [Fagin82 JACM] is one of the following form:

$$(\forall x_1)(\forall x_2)\ldots(\forall x_n)((A_1 \wedge A_2 \wedge \ldots \wedge A_n) \Rightarrow (\exists y_1)(\exists y_2)\ldots(\exists y_r)(B_1 \wedge B_2 \wedge \ldots \wedge B_s))$$

  - Each A_i is a relational atom.
  - Each B_i is a relational atom or an equality.
  - The left-hand side it typed.

Existence of Optimal Insertions

- *Question*: Under what conditions are optimal insertions guaranteed to exist?
- An XEID (extended embedded implicational dependency) [Fagin82 JACM] is one of the following form:

$$(\forall x_1)(\forall x_2)\ldots(\forall x_n)((A_1 \wedge A_2 \wedge \ldots \wedge A_n) \Rightarrow (\exists y_1)(\exists y_2)\ldots(\exists y_r)(B_1 \wedge B_2 \wedge \ldots \wedge B_s))$$

  - Each A_i is a relational atom.
  - Each B_i is a relational atom or an equality.
  - The left-hand side it typed.
- XEIDs subsume virtually all database dependencies which have been studied.

Existence of Optimal Insertions

- *Question*: Under what conditions are optimal insertions guaranteed to exist?
- An XEID (extended embedded implicational dependency) [Fagin82 JACM] is one of the following form:

$$(\forall x_1)(\forall x_2)\ldots(\forall x_n)((A_1 \wedge A_2 \wedge \ldots \wedge A_n) \Rightarrow (\exists y_1)(\exists y_2)\ldots(\exists y_r)(B_1 \wedge B_2 \wedge \ldots \wedge B_s))$$

  - Each A_i is a relational atom.
  - Each B_i is a relational atom or an equality.
  - The left-hand side it typed.
- XEIDs subsume virtually all database dependencies which have been studied.
- They enjoy a key property of *faithfulness* [Fagin82 JACM].

Context:

$\mathbf{D}$ = XEID relational schema $\quad (\mathbf{V}, \gamma : \mathbf{D} \to \mathbf{V})$ = an SPJ view of $\mathbf{D}$

$M_1 \in \mathsf{DB}(\mathbf{D}) \quad (\gamma(M_1), N_2)$ = an insertion on $\mathbf{V}$.

Existence of Optimal Insertions

Context:

$$\mathbf{D} = \text{XEID relational schema} \qquad (\mathbf{V}, \gamma : \mathbf{D} \to \mathbf{V}) = \text{an SPJ view of } \mathbf{D}$$

$$M_1 \in \text{DB}(\mathbf{D}) \qquad (\gamma(M_1), N_2) = \text{an insertion on } \mathbf{V}.$$

Theorem: In the above context, every insertion which is minimal with respect to the information content defined by $\text{WFS}(\mathbf{D}, \exists\wedge+, \text{ConstSym}(M_1))$ is optimal. □

Existence of Optimal Insertions

Context:

$\mathbf{D}$ = XEID relational schema $\quad (\mathbf{V}, \gamma : \mathbf{D} \to \mathbf{V})$ = an SPJ view of $\mathbf{D}$

$M_1 \in \mathrm{DB}(\mathbf{D})$ $\quad (\gamma(M_1), N_2)$ = an insertion on $\mathbf{V}$.

Theorem: In the above context, every insertion which is minimal with respect to the information content defined by $\mathrm{WFS}(\mathbf{D}, \exists\wedge+, \mathrm{ConstSym}(M_1))$ is optimal. □

Corollary: In the above context, all minimal insertions, with respect to the information content defined by $\mathrm{WFS}(\mathbf{D}, \exists\wedge+, \mathrm{ConstSym}(M_1))$, are isomorphic up to a renaming of the newly-introduced constant symbols. □

Existence of Optimal Insertions

Context:

$\mathbf{D}$ = XEID relational schema $(\mathbf{V}, \gamma : \mathbf{D} \to \mathbf{V})$ = an SPJ view of $\mathbf{D}$

$M_1 \in \mathsf{DB}(\mathbf{D})$ $(\gamma(M_1), N_2)$ = an insertion on $\mathbf{V}$.

Theorem: In the above context, every insertion which is minimal with respect to the information content defined by $\mathsf{WFS}(\mathbf{D}, \exists\wedge+, \mathsf{ConstSym}(M_1))$ is optimal. □

Corollary: In the above context, all minimal insertions, with respect to the information content defined by $\mathsf{WFS}(\mathbf{D}, \exists\wedge+, \mathsf{ConstSym}(M_1))$, are isomorphic up to a renaming of the newly-introduced constant symbols. □

- *Question*: When do minimal insertions exist?

Existence of Optimal Insertions

Context:

$\mathbf{D}$ = XEID relational schema $\quad (\mathbf{V}, \gamma : \mathbf{D} \to \mathbf{V})$ = an SPJ view of $\mathbf{D}$

$M_1 \in \mathsf{DB}(\mathbf{D}) \quad (\gamma(M_1), N_2)$ = an insertion on $\mathbf{V}$.

Theorem: In the above context, every insertion which is minimal with respect to the information content defined by $\mathsf{WFS}(\mathbf{D}, \exists\wedge+, \mathsf{ConstSym}(M_1))$ is optimal. $\square$

Corollary: In the above context, all minimal insertions, with respect to the information content defined by $\mathsf{WFS}(\mathbf{D}, \exists\wedge+, \mathsf{ConstSym}(M_1))$, are isomorphic up to a renaming of the newly-introduced constant symbols. $\square$

- *Question*: When do minimal insertions exist?
- *Answer*: Not always, the chase procedure is required to terminate.

Existence of Optimal Insertions

Context:

$\mathbf{D}$ = XEID relational schema $\quad (\mathbf{V}, \gamma : \mathbf{D} \to \mathbf{V})$ = an SPJ view of $\mathbf{D}$

$M_1 \in \mathsf{DB}(\mathbf{D}) \quad (\gamma(M_1), N_2)$ = an insertion on $\mathbf{V}$.

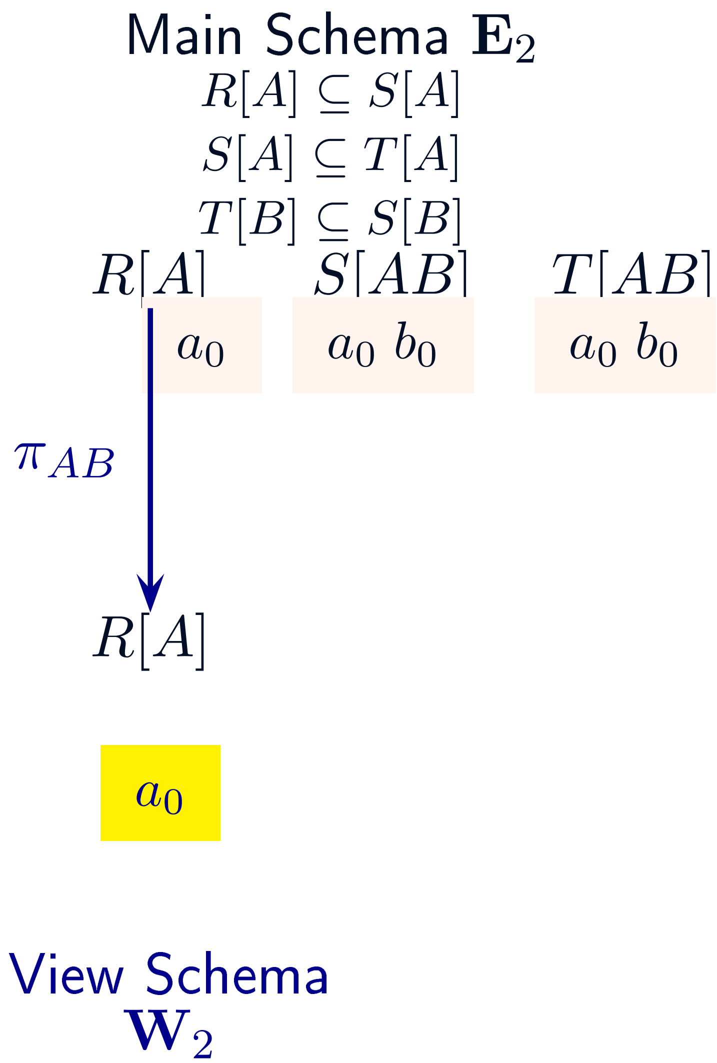
Theorem: In the above context, every insertion which is minimal with respect to the information content defined by $\mathsf{WFS}(\mathbf{D}, \exists\wedge+, \mathsf{ConstSym}(M_1))$ is optimal. □

Corollary: In the above context, all minimal insertions, with respect to the information content defined by $\mathsf{WFS}(\mathbf{D}, \exists\wedge+, \mathsf{ConstSym}(M_1))$, are isomorphic up to a renaming of the newly-introduced constant symbols. □

- *Question*: When do minimal insertions exist?
- *Answer*: Not always, the chase procedure is required to terminate.
- This may be guaranteed by restricting attention to the *weakly acyclic* dependencies [Fagin et al 2005].

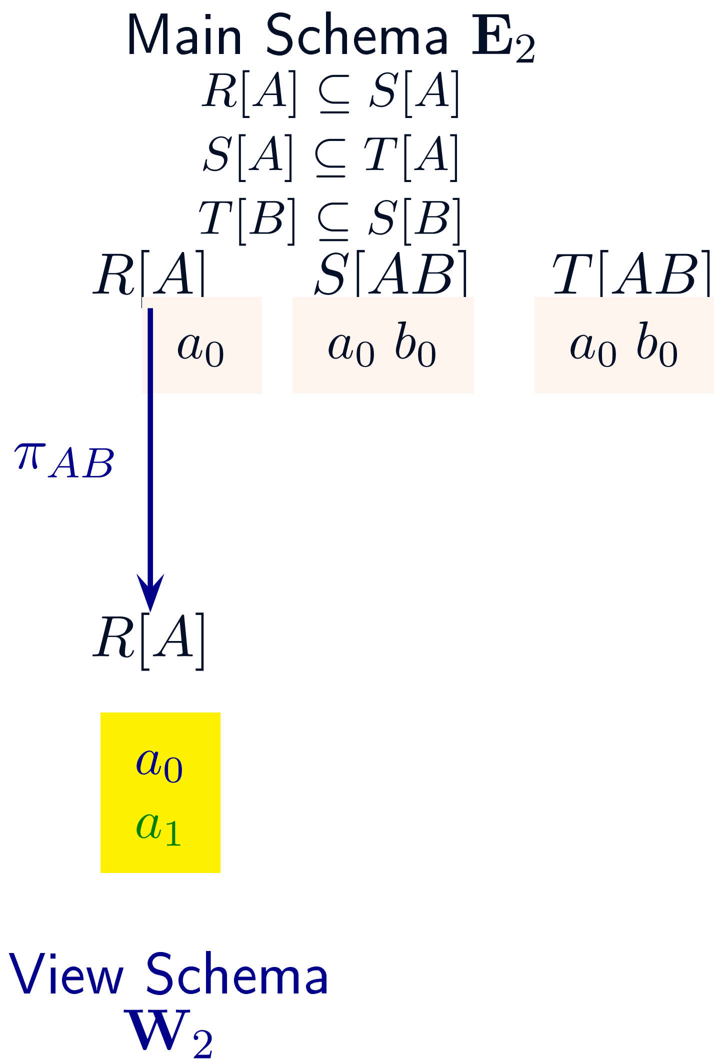
Example of Non-Existence of Optimal Insertions

- The schema and initial states are shown.



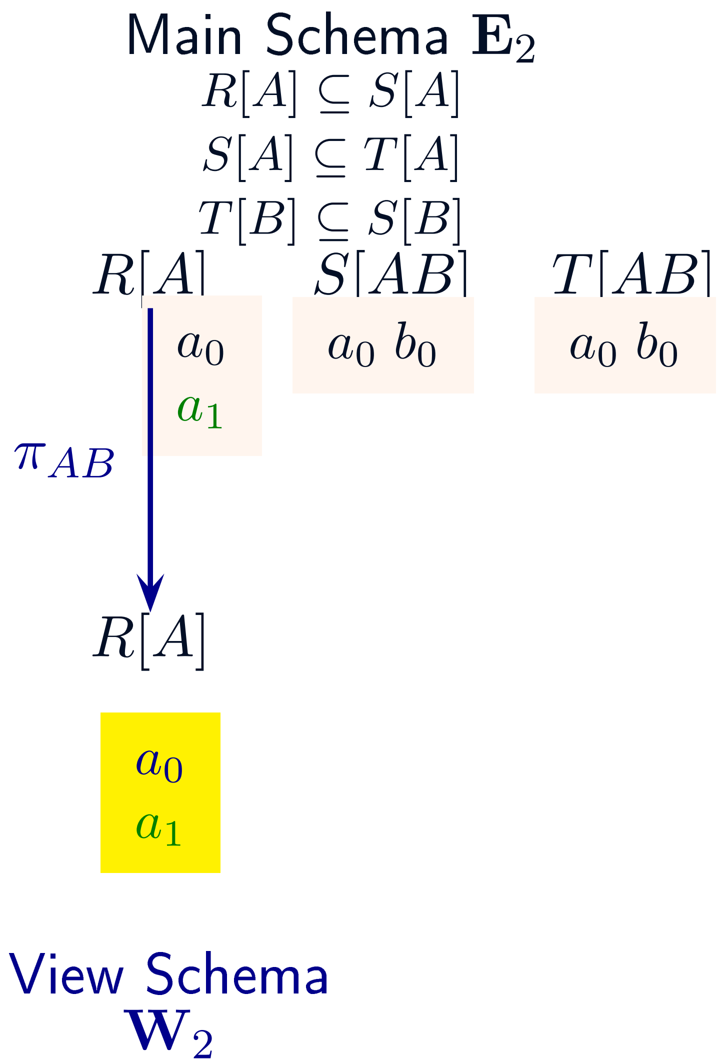
Example of Non-Existence of Optimal Insertions

- The schema and initial states are shown.
- Consider the view update Insert$\langle R(a_1)\rangle$.



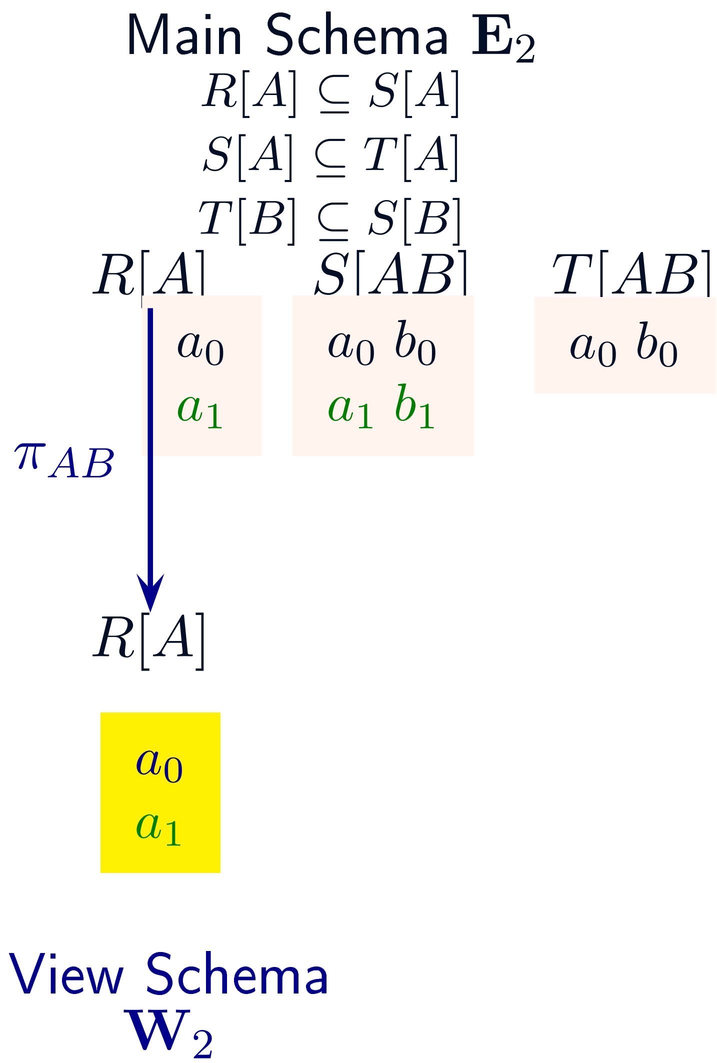
Example of Non-Existence of Optimal Insertions

- The schema and initial states are shown.
- Consider the view update Insert$\langle R(a_1) \rangle$.



Example of Non-Existence of Optimal Insertions

- The schema and initial states are shown.
- Consider the view update Insert$\langle R(a_1) \rangle$.



Example of Non-Existence of Optimal Insertions

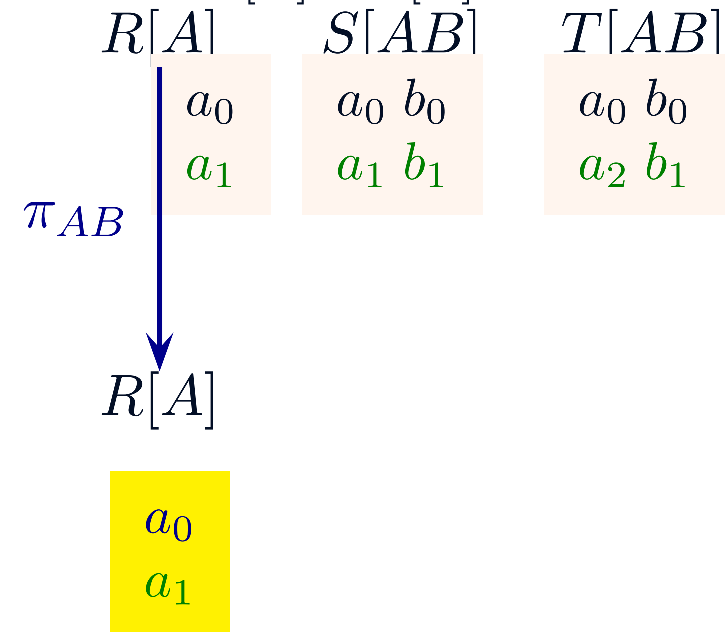
- The schema and initial states are shown.
- Consider the view update Insert$\langle R(a_1)\rangle$.

Main Schema $\mathbf{E}_2$

$R[A] \subseteq S[A]$

$S[A] \subseteq T[A]$

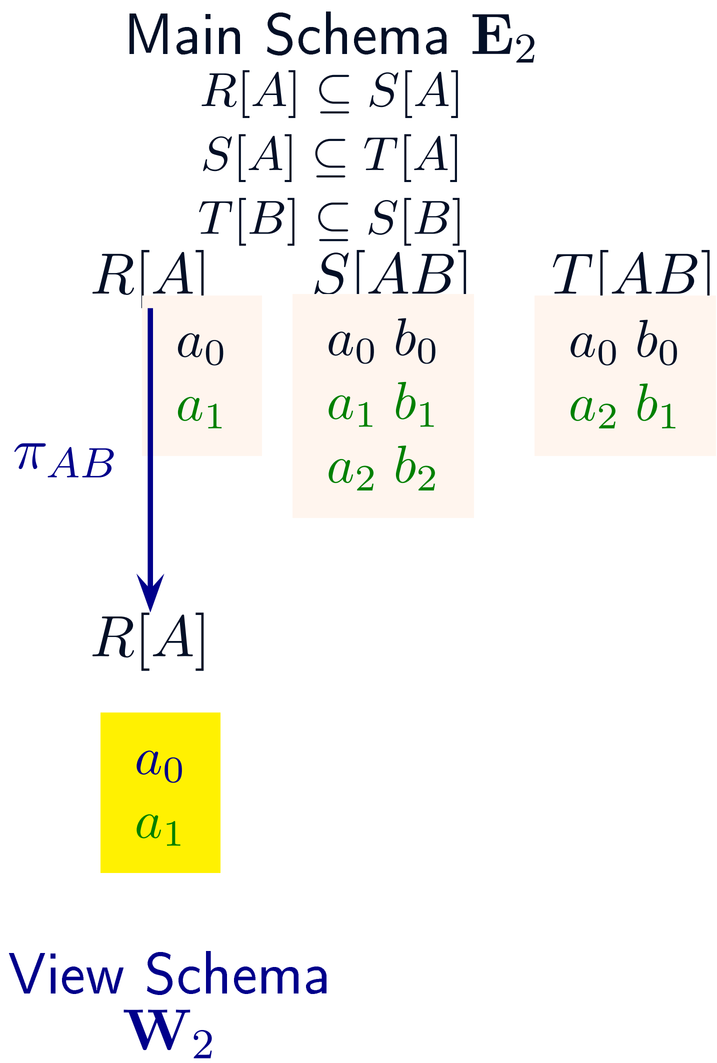
$T[B] \subseteq S[B]$



View Schema $\mathbf{W}_2$

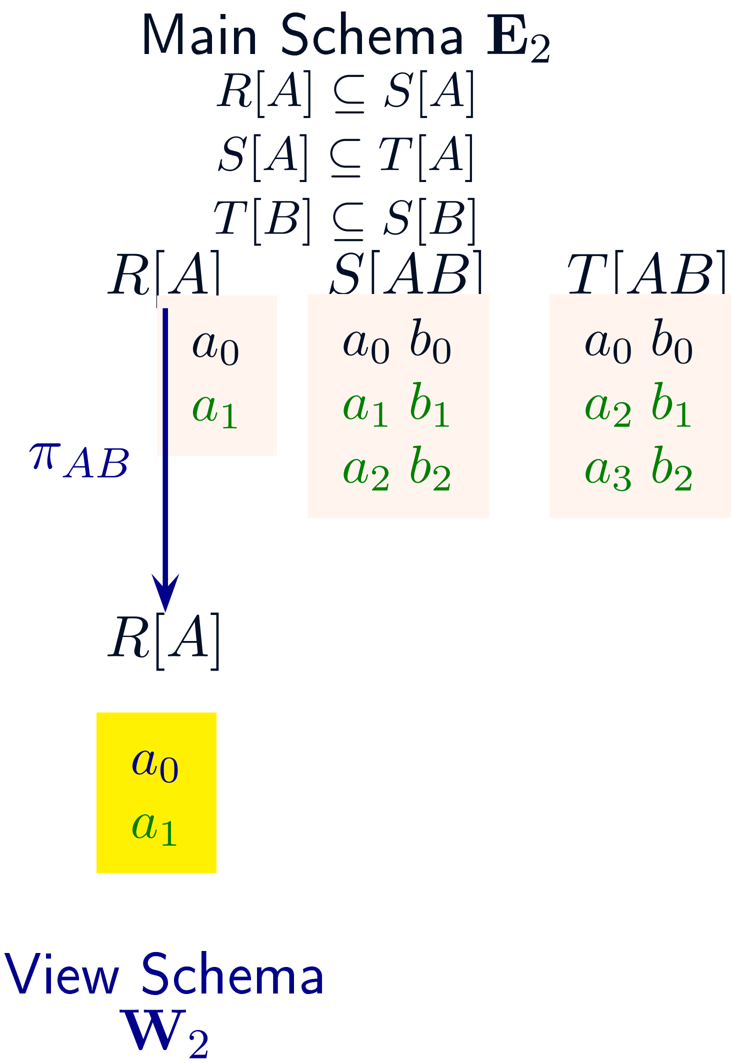
Example of Non-Existence of Optimal Insertions

- The schema and initial states are shown.
- Consider the view update Insert$\langle R(a_1) \rangle$.

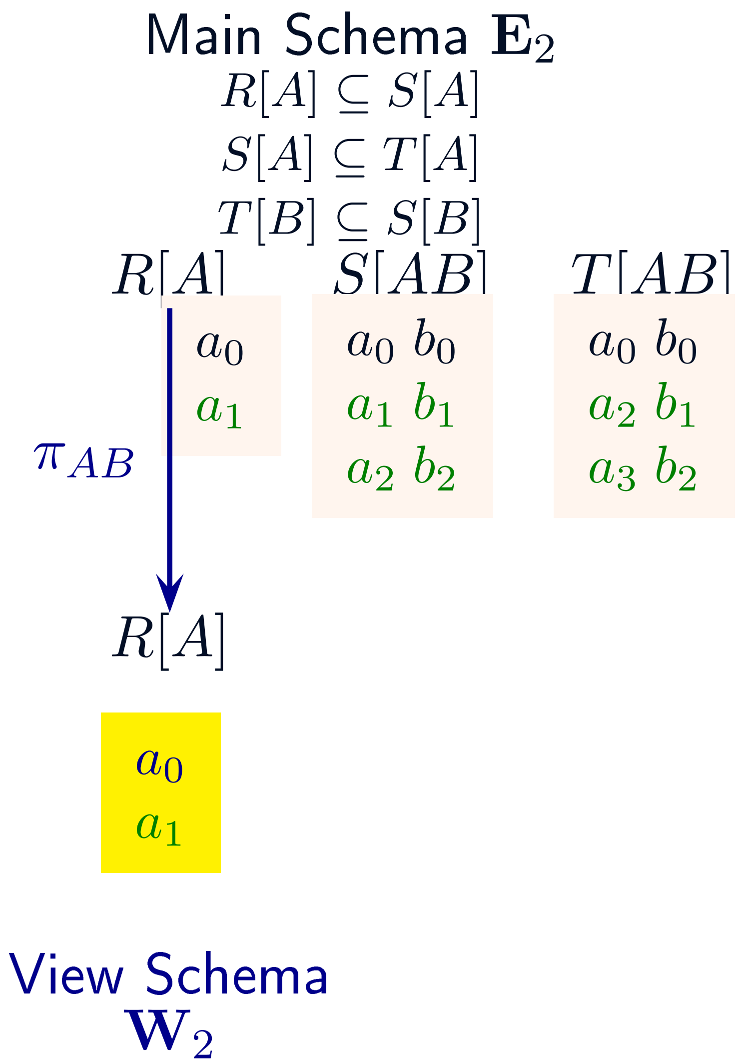


Example of Non-Existence of Optimal Insertions

- The schema and initial states are shown.
- Consider the view update Insert$\langle R(a_1)\rangle$.
- This process continues endlessly.

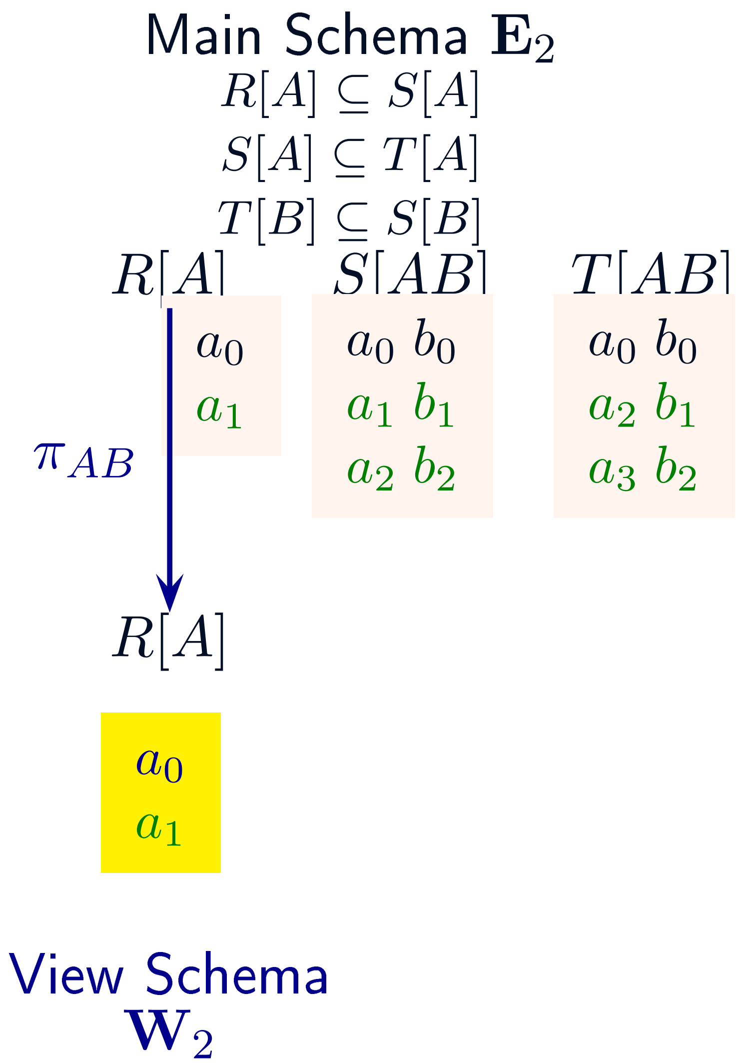


- The schema and initial states are shown.
- Consider the view update Insert$\langle R(a_1)\rangle$.
- This process continues endlessly.
- A decision to reuse an existing value must be made.



Example of Non-Existence of Optimal Insertions

- The schema and initial states are shown.
- Consider the view update Insert$\langle R(a_1) \rangle$.
- This process continues endlessly.
- A decision to reuse an existing value must be made.
- However, such a decision clearly leads to sub-optimality.



Conclusions and Properties of the Solution Technique

- A logic-based technique for measuring the quality of a reflection of a view update has been presented.

Conclusions and Properties of the Solution Technique

- A logic-based technique for measuring the quality of a reflection of a view update has been presented.
- This technique is strictly finer grained than simply counting the number of tuples which change.

Conclusions and Properties of the Solution Technique

- A logic-based technique for measuring the quality of a reflection of a view update has been presented.
- This technique is strictly finer grained than simply counting the number of tuples which change.
- Under common conditions, it has been shown that all optimal updates are isomorphic up to a renaming of the new constant symbols which are introduced.

Conclusions and Properties of the Solution Technique

- A logic-based technique for measuring the quality of a reflection of a view update has been presented.
- This technique is strictly finer grained than simply counting the number of tuples which change.
- Under common conditions, it has been shown that all optimal updates are isomorphic up to a renaming of the new constant symbols which are introduced.
- When the main schema is constrained by XEIDs and the view is SPJ, all optimal solutions are isomorphic.

Conclusions and Properties of the Solution Technique

- A logic-based technique for measuring the quality of a reflection of a view update has been presented.
- This technique is strictly finer grained than simply counting the number of tuples which change.
- Under common conditions, it has been shown that all optimal updates are isomorphic up to a renaming of the new constant symbols which are introduced.
- When the main schema is constrained by XEIDs and the view is SPJ, all optimal solutions are isomorphic.
- Optimal solutions exist in case the chase inference procedure terminates.

Optimization of tuple modification:

Further Directions

Optimization of tuple modification:

- The existing approach focuses upon insertions.

Optimization of tuple modification:

- The existing approach focuses upon insertions.
- Deletions are an easy extension.

Further Directions

Optimization of tuple modification:

- The existing approach focuses upon insertions.
- Deletions are an easy extension.
- Modifications almost never have an optimal solution, because they cannot distinguish a change from a combination of insertions and deletions.

Optimization of tuple modification:

- The existing approach focuses upon insertions.
- Deletions are an easy extension.
- Modifications almost never have an optimal solution, because they cannot distinguish a change from a combination of insertions and deletions.
- An alternative model is necessary for this case.

Further Directions

Optimization of tuple modification:

- The existing approach focuses upon insertions.
- Deletions are an easy extension.
- Modifications almost never have an optimal solution, because they cannot distinguish a change from a combination of insertions and deletions.
- An alternative model is necessary for this case.

Application to database components:

Further Directions

Optimization of tuple modification:

- The existing approach focuses upon insertions.
- Deletions are an easy extension.
- Modifications almost never have an optimal solution, because they cannot distinguish a change from a combination of insertions and deletions.
- An alternative model is necessary for this case.

Application to database components:

- Cooperative updates to database components has been studied [Hegner & Schmidt 2007 ADBIS]

Further Directions

Optimization of tuple modification:

- The existing approach focuses upon insertions.
- Deletions are an easy extension.
- Modifications almost never have an optimal solution, because they cannot distinguish a change from a combination of insertions and deletions.
- An alternative model is necessary for this case.

Application to database components:

- Cooperative updates to database components has been studied [Hegner & Schmidt 2007 ADBIS]
- Methods which combine cooperative update with the automated choices of this paper deserve further investigation.

Further Directions

Optimization of tuple modification:

- The existing approach focuses upon insertions.
- Deletions are an easy extension.
- Modifications almost never have an optimal solution, because they cannot distinguish a change from a combination of insertions and deletions.
- An alternative model is necessary for this case.

Application to database components:

- Cooperative updates to database components has been studied [Hegner & Schmidt 2007 ADBIS]
- Methods which combine cooperative update with the automated choices of this paper deserve further investigation.

Relationship to work in logic programming: