Transaction Systems
Exercise Session 03

Linnea Passing
linnea.passing@tum.de
www-db.in.tum.de

May 02, 2016
Today’s Plan

- Last week’s homework
- Questions
- FSR, VSR, CSR, OCSR
- Homework
Questions

- Is LRF not considering the written variable?
- What does “subsequent” mean in the definition of direct usefulness?
- Is the positioning of commits important for serializability?
Questions

- Is LRF not considering the written variable?
  - Yes, only the second transaction (read) must be alive
- What does “subsequent” mean in the definition of direct usefulness?
- Is the positioning of commits important for serializability?
Questions

- Is LRF not considering the written variable?
  - Yes, only the second transaction (read) must be alive
- What does “subsequent” mean in the definition of direct usefulness?
  - Afterwards, later; no further restriction
- Is the positioning of commits important for serializability?
Questions

- Is LRF not considering the written variable?
  - Yes, only the second transaction (read) must be alive

- What does “subsequent” mean in the definition of direct usefulness?
  - Afterwards, later; no further restriction

- Is the positioning of commits important for serializability?
  - No, because we are only looking at committing transactions anyway (cf. monotony!)
FSR, VSR, CSR, OCSR

- Credits: Dr. Andrey Gubichev, 2013
The reads-from relation of $s$:

$$RF(s) = \{(t_i, x, t_j) \mid r_j(x) \text{ reads } x \text{ from } w_i(x)\}$$
Alive steps

- Step $p$ is directly useful for step $q$ ($p \rightarrow q$), if
  - $q$ reads from $p$
  - or $p$ is read and $q$ is subsequent write in the same transaction
- Transitive closure of $\rightarrow$: $\ast \rightarrow$
- A step $p$ is alive if it is useful for some step from $t_\infty$
Example

\[ s = r_1(x)r_2(y)w_1(y)w_2(y)c_1c_2r_\infty(x)r_\infty(y)c_\infty \]
Example

\[ s = r_1(x)r_2(y)w_1(y)w_2(y)c_1c_2r_\infty(x)r_\infty(y)c_\infty \]

- \( w_2(y) \) is alive
- \( r_2(y) \) is alive
- \( r_1(x) \) is not alive
Live Reads-from

The *live reads-from* relation of $s$:

$$LRF(s) = \{(t_i, x, t_j) \mid \text{an alive } r_j(x) \text{ reads } x \text{ from } w_i(x)\}$$
Example

\[ s = w_0(x)w_0(y)c_0r_1(x)r_2(y)w_1(y)w_2(y)c_1c_2r_\infty(x)r_\infty(y)c_\infty \]
Example

\[ s = w_0(x)w_0(y)c_0 r_1(x)r_2(y)w_1(y)w_2(y)c_1 c_2 r_\infty(x)r_\infty(y)c_\infty \]

\[ RF(s) = \{(t_0, x, t_1), (t_0, y, t_2), (t_0, x, t_\infty), (t_2, y, t_\infty)\} \]

\[ LRF(s) = \{(t_0, y, t_2), (t_0, x, t_\infty), (t_2, y, t_\infty)\} \]
Two schedules are FSR equivalent iff

\[ \begin{align*}
\text{op}(s) &= \text{op}(s') \\
\text{LRF}(s) &= \text{LRF}(s')
\end{align*} \]
Two schedules are *view equivalent*, iff

- $\text{op}(s) = \text{op}(s')$
- $H[s] = H[s']$
- $H_s(p) = H_{s'}(p)$ for every read or write step $p$
Two schedules are *view equivalent*, iff

- $\text{op}(s) = \text{op}(s')$
- $\text{RF}(s) = \text{RF}(s')$
Example

\[ s = w_0(x)w_0(y)c_0r_1(x)w_1(x)r_2(x)r_2(y)w_2(y)c_2w_1(y)c_1 \]
Example

\[ s = w_0(x)w_0(y)c_0 r_1(x)w_1(x)r_2(x)r_2(y)w_2(y)c_2 w_1(y)c_1 \]

- \( RF(s) = \{(t_0, x, t_1), (t_1, x, t_2), (t_0, y, t_2), (t_1, x, t_\infty), (t_1, y, t_\infty)\} \)
- \( LRF(s) = \{(t_0, x, t_1), (t_1, x, t_\infty), (t_1, y, t_\infty)\} \)
Example

\[ s = w_0(x)w_0(y)c_0r_1(x)w_1(x)r_2(x)r_2(y)w_2(y)c_2w_1(y)c_1 \]

\[ RF(s) = \left\{ (t_0, x, t_1), (t_1, x, t_2), (t_0, y, t_2), (t_1, x, t_\infty), (t_1, y, t_\infty) \right\} \]

\[ LRF(s) = \left\{ (t_0, x, t_1), (t_1, x, t_\infty), (t_1, y, t_\infty) \right\} \]

\[ \text{not in VSR (but in FSR)} \]
Order preservation

A history $s$ is order-preserved conflict serializable, iff there is a serial $s'$:

- $\text{op}(s) = \text{op}(s')$
- $s \approx_c s'$
- For all $t, t'$: if $t$ occurs completely before $t'$ in $s$, the same holds for $s'$
Example

\[ s = r_1(x)w_1(z)r_2(z)w_1(y)c_1 r_3(y)w_2(z)c_2 w_3(x)w_3(y)c_3 \]
Projection of a schedule

- $s$ is a schedule, $T \subseteq \text{trans}(s)$.
- A Projection $\Pi_T(s)$ of $s$ onto $T$ is a result of erasing of all steps of transactions not in $T$.
- $s = w_1(x)r_2(x)w_2(y)r_1(y)w_1(y)w_3(x)w_3(y)c_1a_2$
- $T = \{t_1, t_2\}$. 
Projection of a schedule

- $s$ is a schedule, $T \subseteq \text{trans}(s)$.
- A Projection $\Pi_T(s)$ of $s$ onto $T$ is a result of erasing of all steps of transactions not in $T$
- $s = w_1(x)r_2(x)w_2(y)r_1(y)w_1(y)w_3(x)w_3(y)c_1a_2$
- $T = \{t_1, t_2\}$. $\Pi_T(s) = w_1(x)r_2(x)w_2(y)r_1(y)w_1(y)c_1a_2$
Monotone class of histories

A class $E$ of histories is monotone if:

- if $s \in E$: for each $T \subseteq \text{trans}(s)$ holds $\prod_T(s) \in E$
A class $E$ of histories is monotone if:

- if $s \in E$: for each $T \subseteq \text{trans}(s)$ holds $\prod_T(s) \in E$

VSR is not monotone:

$$s = w_1(x)w_2(x)w_2(y)c_2w_1(y)c_1w_3(x)w_3(y)c_3$$
Schedulers

We already have seen:

- 2PL
- Strict 2PL
- Strong 2PL
- $s = r_1(x)r_3(y)w_3(y)r_2(z)w_2(x)r_4(y)c_3w_4(z)c_4c_2c_1$
FSR, VSR, CSR, OCSR

- Credits: Dr. Andrey Gubichev, 2013
Homework

- Already uploaded to our website.