Layered locking by example

- locks for high level operations
- page-level locks held only during corresponding high level operation
- this schedule would not be possible in page model
Layered locking: formally

- Lock acquisition: acquire $L_i$ lock on $x$ before the $f(x)$ can start at level $L_i$
- Lock release: once the $L_i$ lock is released, no other child of this subtransaction can get any lock
- Subtransaction: once the operation $f(x)$ of $L_i$ is finished, all locks at $L_{i-1}$ for the children of $f(x)$ are released
- Modular-based lock manager: for every level $L_i$
Layered locking: deadlocks

- No "global" deadlocks (between different layers)
- "Local" deadlocks: roll back the operation that causes deadlock
Layered locking: selective layered 2PL

- choose some layers, skip all the rest
- skip the layer $L_i$ by extending the scope of subtransactions above $L_i$
- less lock management overhead, but also less concurrency
Selective Layered 2PL Example

Insert Into Persons
Values (Name=..., City="Austin", Age=29, ...)

Select Name From Persons
Where City="Seattle" And Age=29

Select Name From Persons
Where Age=30
Hybrid protocols

- Idea of modularity: for every level use its own protocol
- Why? E.g., the fraction of read-only operation at a page level is higher than the fraction of read-only operations at the root level
- 2PL at object level, FOCC at the page level
- 2PL at $L_1$ and ROMV at $L_0$
Escrow locking

- For counter objects: there are bounds low, high and current possible value range sup, inf

- incr (x, D):
  
  if x.sup + D <= x.high then
    x.sup := x.sup + D; return ok
  
  else if x.inf + D > x.high then
    return no

  else wait fi fi;

- decr (x, D):
  
  if x.low <= x.inf - D then
    x.inf := x.inf - D; return ok

  else if x.low > x.sup - D then
    return no

  else wait fi fi;
Escrow locking

- When committing: adjust $inf$, $sup$
- When aborting: roll back $inf$, $sup$

Commit of transaction $t$:
for each operation $\text{incr}(x, D)$ executed by $t$ do
\[
x.\text{inf} := x.\text{inf} + D
\]
od;
for each operation $\text{decr}(x, D)$ executed by $t$ do
\[
x.\text{sup} := x.\text{sup} - D
\]
od;

Abort of transaction $t$:
for each operation $\text{incr}(x, D)$ executed by $t$ do
\[
x.\text{sup} := x.\text{sup} - D
\]
od;
for each operation $\text{decr}(x, D)$ executed by $t$ do
\[
x.\text{inf} := x.\text{inf} + D
\]
Example

Consider two counter objects \( x \) and \( y \), with initial values \( x = 100 \) and \( y = 50 \). Both counters have zero as a lower bound and no upper bound.

\[
de\!cr_1(x, 60)\ incr_2(x, 20)\ incr_1(x, 10)\ de\!cr_3(x, 50)\ de\!cr_2(y, 60)\
\ incr_2(x, 20)\ a_2\ de\!cr_1(y, 10)\ c_1\ c_3
\]
Homework: Task 1

- Apply the following hybrid protocols to the schedule (see next slide) in a *selective way*, such that only two levels are involved:
  - forward-oriented optimistic concurrency control (FOCC) at the page level and strong 2PL at the record level
  - FOCC at the page level, strong 2PL at the query level
  - ROMV at the page level, strong 2PL at the record level
  - ROMV at the page level, strong 2PL at the query level
3-Level Example

```
Insert Into Persons
Values (Name=..., City="Austin", Age=29, ...)

Select Name
From Persons
Where City="Seattle"
And Age=29

Select Name
From Persons
Where Age=30
```
Consider the following transactions:

\[ t_1 = r_1(x)w_1(x)r_1(y)w_1(y) \]

\[ t_2 = r_2(x) \]

\[ t_3 = r_3(y)w_3(y) \]

Try to decompose \( t_1 \) into three pieces such that the result is a correct chopping.
Info

- Exercises due: 9 AM, January 20, 2014
- Submit to andrey.gubichev@in.tum.de