updates 2.0
prof. Stratos Idreos

HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS165/
**UPDATE** table_name
**SET** column1=value1,column2=value2,...
**WHERE** some_column=some_value

**INSERT INTO** table_name
**VALUES** (value1,value2,value3,...)
the world has changed a little bit by now...
monitor CPU utilization

monitor memory hierarchy utilization

monitor clicks
(frequency, locations, specific links, sequences)
insert new entry (a,b,c,d,...) on table x
update N columns, K trees, statistics, ...

Table x

A  B  C  D

...
A case for fractured mirrors
Ravishankar Ramamurthy, David J. DeWitt, Qi Su
Very Large Databases Journal (VLDBJ), 2003
in-place updates: the cardinal sin

The Transaction Concept: Virtues and Limitations
Jim Gray, Tandem TR 81.3, 1981
what can go wrong?
failures & >1 queries
update all rows where A=v1 & B=v2 to (a=a/2, b=b/4, c=c-3, d=d+2)
update all rows
where A=v1 & B=v2
to (a=a/2, b=b/4, c=c-3, d=d+2)
update all rows where $A=v_1$ & $B=v_2$ to $(a=a/2, b=b/4, c=c-3, d=d+2)$

list of rowIDs (positions) (sort)

search (scan/index) to find row to update

select+project actions

(a=a/2, b=b/4, c=c-3, d=d+2)
update all rows where $A=v_1 \land B=v_2$
to $(a=a/2, b=b/4, c=c-3, d=d+2)$

we know what to update but nothing happened yet
update all rows where \(A=v_1 \& B=v_2\) to \((a=a/2,b=b/4,c=c-3,d=d+2)\)

WAL: keep persistent notes as we go so we can resume or undo

if problem (power/abort) before we write all pages we are left with an inconsistent state
update all rows where $A=v_1 \& B=v_2$ to $(a=a/2, b=b/4, c=c-3, d=d+2)$
update all rows where $A=v_1 \& B=v_2$ to $(a=a/2, b=b/4, c=c-3, d=d+2)$
update all rows where $A=v_1 \& B=v_2$ to $(a=a/2, b=b/4, c=c-3, d=d+2)$
update all rows where \( A=v_1 \) & \( B=v_2 \) to \( (a=a/2, b=b/4, c=c-3, d=d+2) \)

![Diagram showing level 1 and level 2 with CPU labeled and update process described](image_url)
update all rows where A=v1 & B=v2 to (a=a/2, b=b/4, c=c-3, d=d+2)

CPU

level 1

level 2

N updates in persistent storage

apply all updates for this page

read page in L1
update persist to L2
update all rows where $A=v1 \& B=v2$ to $(a=a/2, b=b/4, c=c-3, d=d+2)$
**classic example**

Joe owes Mike $100.

Both Joe and Mike have a Bank of Bla account.

Possible actions:
- Joe: $-100
- Mike: $+100
- Joe: $-100
- Mike: $+100

What if there is a failure here?
classic example

joe owes mike 100$

both joe and mike have a Bank of Bla account

possible actions

joe -100 → mike + 100

mike + 100 → joe - 100

what if there is a failure here?

actually
1) read mike; 2) mike + 100; 3) write new mike;
update all rows where \( A=v_1 \) \& \( B=v_2 \) to \((a=a/2, b=b/4, c=c-3, d=d+2)\)

![Diagram]

- **CPU**
- **Level 1**
- **Level 2**

- Read page in L1
- Update
- Persist to L2

What do we need to remember (log)?
update all rows where $A=v_1 \land B=v_2$ to $(a=a/2, b=b/4, c=c-3, d=d+2)$

search (scan/index) to find row to update

select+project actions

list of rowIDs (positions) (sort)

restart: get set of positions again, and either resume from last written page or undo all previously written pages
buffer updates

memory

log

disk

buffer updates

memory

log

SSD

disk

buffer updates

memory

log

non volatile memory

SSD

disk
flash disks
read/write asymmetry

need to erase a block to write it

finite number of erase cycles

MaSM: efficient online updates in data warehouses
Manos Athanassoulis, Shimin Chen, Anastasia Ailamaki, Phillip Gibbons, Radu Stoica
ACM SIGMOD Conference, 2011
ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging
C. Mohan, Donald J. Haderle, Bruce G. Lindsay, Hamid Pirahesh, Peter M. Schwarz
ACM Transactions on Database Systems, 1992

C Mohan, IBM Research
ACM SIGMOD Edgar F. Codd Innovations award 1993
(rumor has it he has his own Facebook server)
(also check his noSQL lecture)
classic example
joe owes mike 100 $
both joe and mike have a Bank of Bla account

what about logging and recovery during e-shopping
e.g., shopping cart, wish list, check out

Quantifying eventual consistency with PBS
Peter Bailis, Shivaram Venkataraman, Michael J. Franklin,
Joseph M. Hellerstein, Ion Stoica
Communications of the ACM, 2014
why noSQL

- complex legacy tuning expensive...

- simple clean just enough ...

as apps become more complex

as apps need to be more scalable

newSQL

Mesa: Geo-Replicated, Near Real-Time, Scalable Data Warehousing
Ashish Gupta et al.
International Conference on Very Large Databases (VLDB), 2014

F1: A Distributed SQL Database That Scales
Jeff Shute et al.
International Conference on Very Large Databases (VLDB), 2013

(more in 265)
all or nothing

remember what we changed so we can undo or resume
(depends on applications semantics)
transaction
any database query that
should be seen as a single task

```
parse
optimize
plan(
read X
write Z
... )
```
Atomicity

Consistency

Isolation

Durability

Jim Gray, IBM, Tandem, DEC, Microsoft
ACM Turing award
ACM SIGMOD Edgar F. Codd Innovations award
Atomicity
Consistency
Isolation
Durability

all or nothing
correct
>1 queries
persistent
queries arrive concurrently
q1) select A>v1, B=B+b, C=C-c
q2) select B>v2, avg(C)
q3) select C<v3 & B>v4, A=C
q4) write more queries

Goal:
process queries in parallel
leave db in a consistent state
return correct results
query 1
parse
optimize

plan(
read X
write Z
... )

query 2
parse
optimize

plan(
read Z
write Z
... )
locks

wait until lock is free

... read X → ... get_rlock(X)
... read X
... release_rlock(X)
... commit

... write X → ... get_wlock(X)
... write X
... release_wlock(X)
... commit

<table>
<thead>
<tr>
<th></th>
<th>R</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>W</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
2 phase locking

1. get all locks
2. do all tasks
3. commit
4. release all locks
(and variations)

be positive! - optimistic concurrency control
A survey of B-tree logging and recovery techniques
Goetz Graefe
textbook: chapters 16, 17, 18
updates 2.0
DATA SYSTEMS
prof. Stratos Idreos