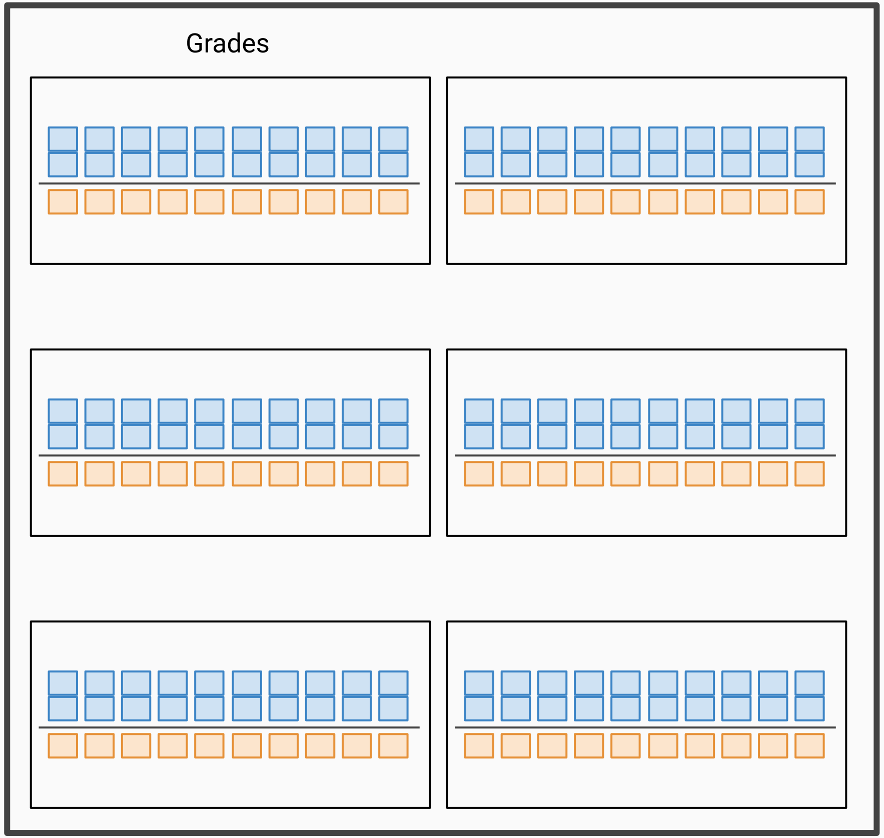
Milestone 2 Presentation

By The Database Daddies

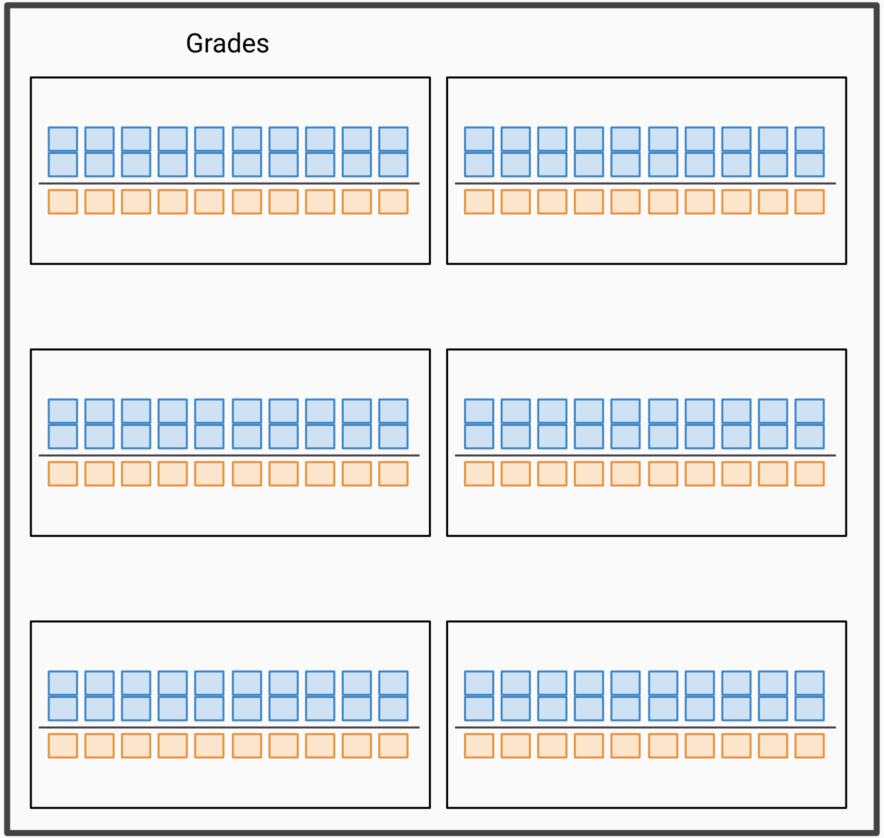
Review: our database

Database is made up of tables

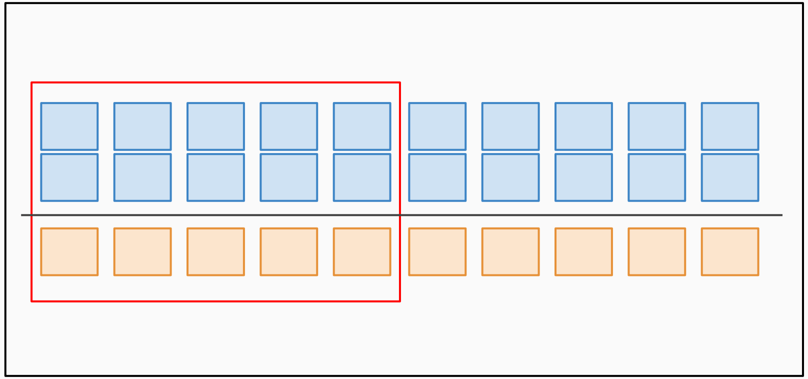


Review: our database

Database is made up of tables

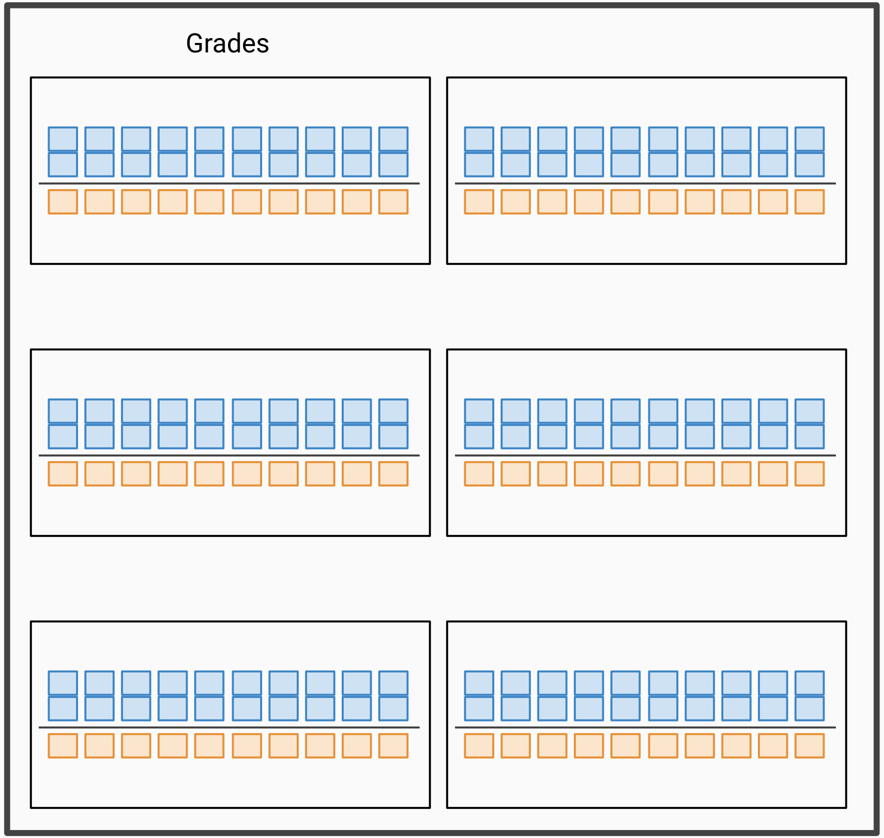


Tables are made up of pages that are organized a certain way

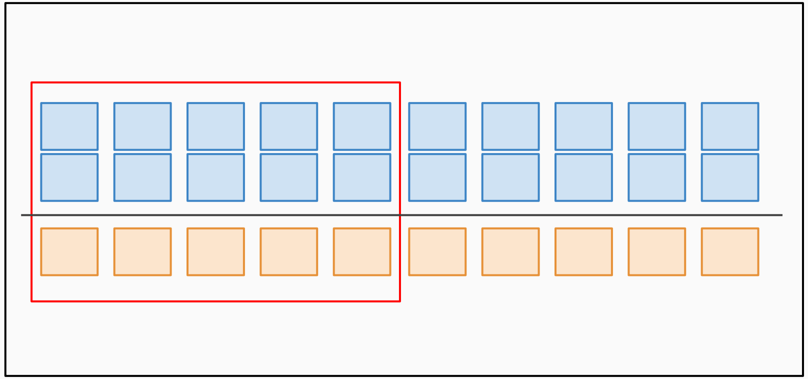


Review: our database

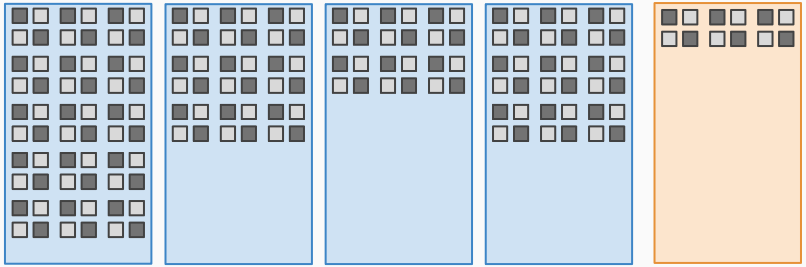
Database is made up of tables



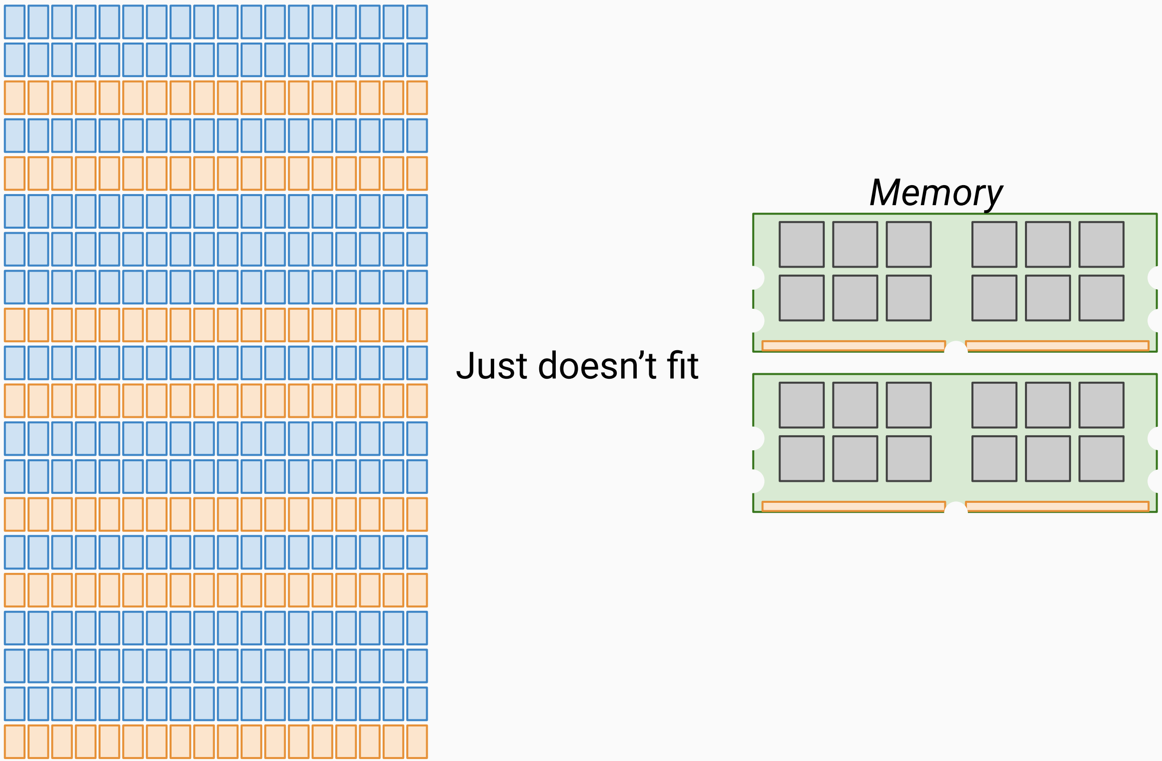
Tables are made up of pages that are organized a certain way



Pages hold bytes ■ □ = Bytes *(your data)*



Bufferpool Management



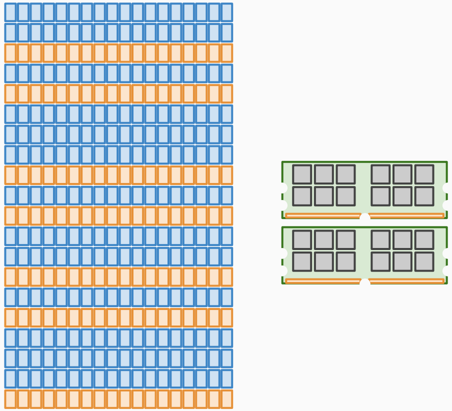
Your Database

Just doesn’t fit

Memory

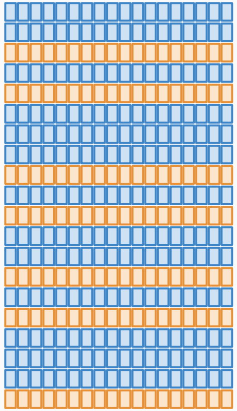
Durability and Bufferpool: The Problem

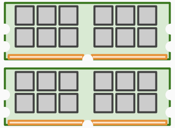
But it does Fit in your SSD
(long term memory)

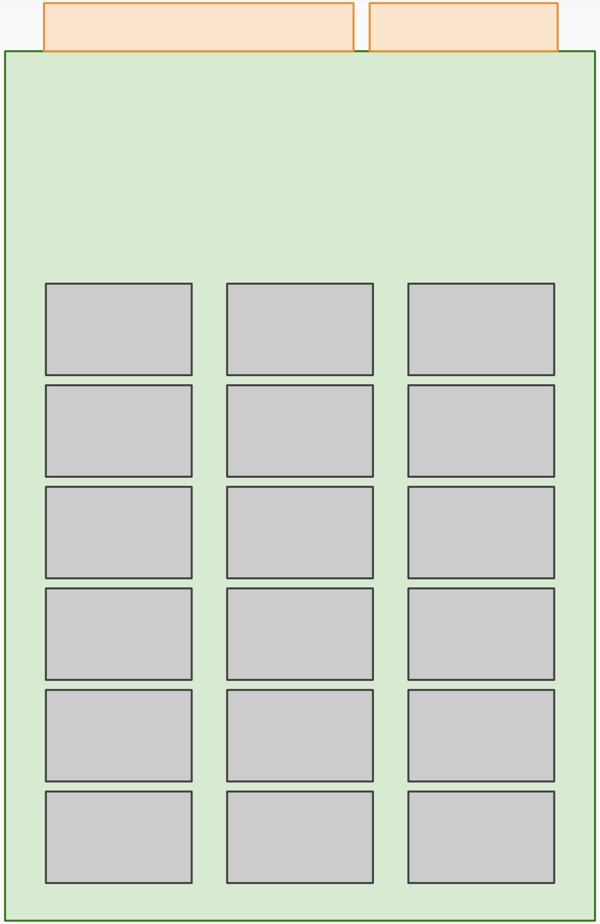


Durability and Bufferpool: The Problem

But it does Fit in your SSD
(long term memory)

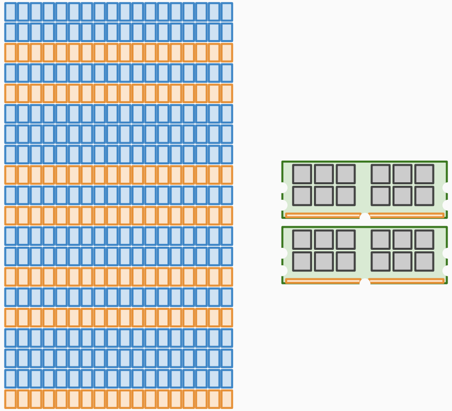




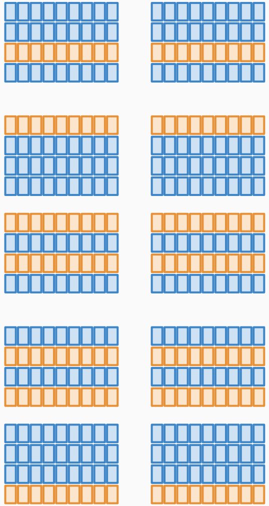


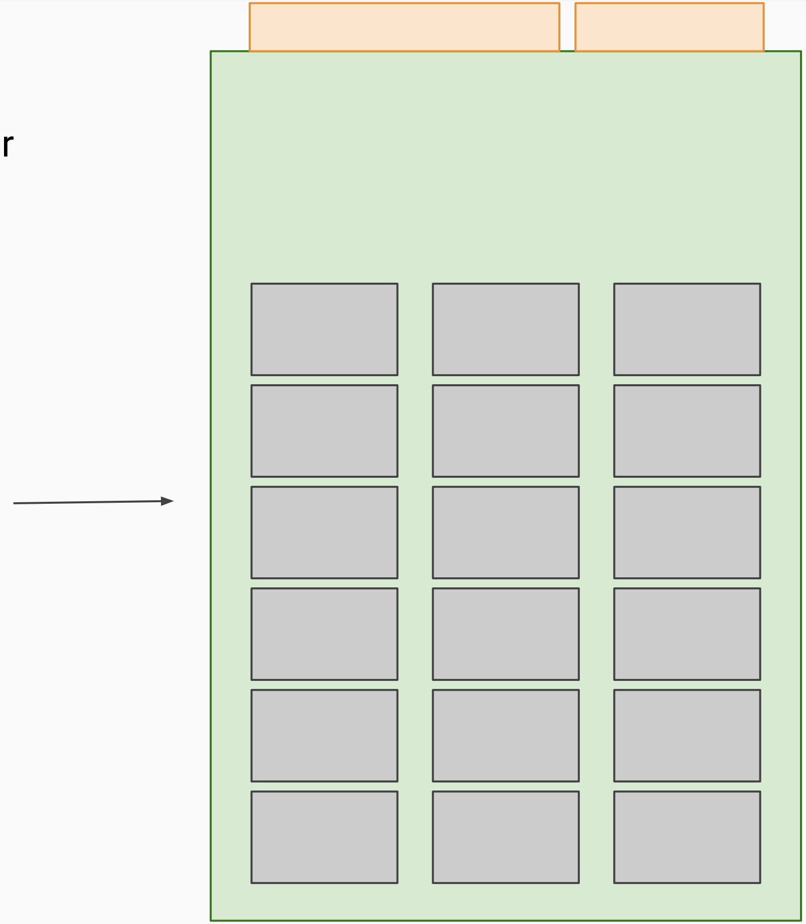
Durability and Bufferpool: The Problem

But it does Fit in your SSD
(long term memory)



(These objects are
going to be used for
animation later)





Durability and Bufferpool: Solution

Bufferpools!

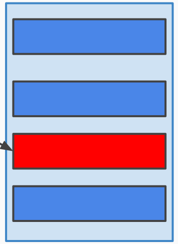
This slide should explain the idea of the bufferpool and how we implemented it

- What is the bufferpool?
- What is our implementation of bufferpool?

Durability and Bufferpool Extension Overview

The Bufferpool Design: Assumes that we have a pool of memory pages for reading and writing data from and to persistent storage(disk).

- Dirty Page

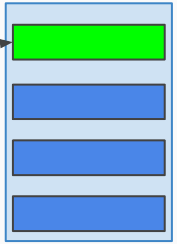


When page is updated, it's marked as dirty by transaction.

Bufferpool keeps track of dirty pages to flush back to disk after replacement or when close() is invoked.

- Pinning and Unpinning

Page is accessed by one or more transactions

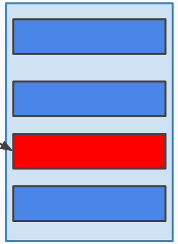


Will be pinned while accessed. And will be unpinned once transactions no longer need the page.

Durability and Bufferpool Extension Implementation

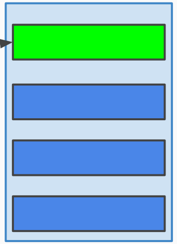
Bufferpool Eviction Policy:

- Dirty Page

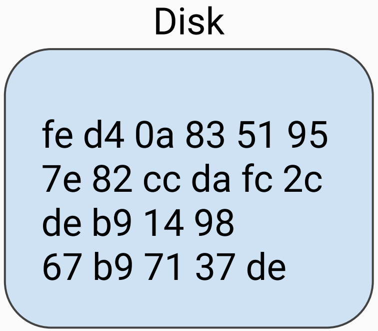


- Pinning and Unpinning

Page is accessed by one or more transactions



Data is stored on to a disk by persisting our data in/out of a binary file.

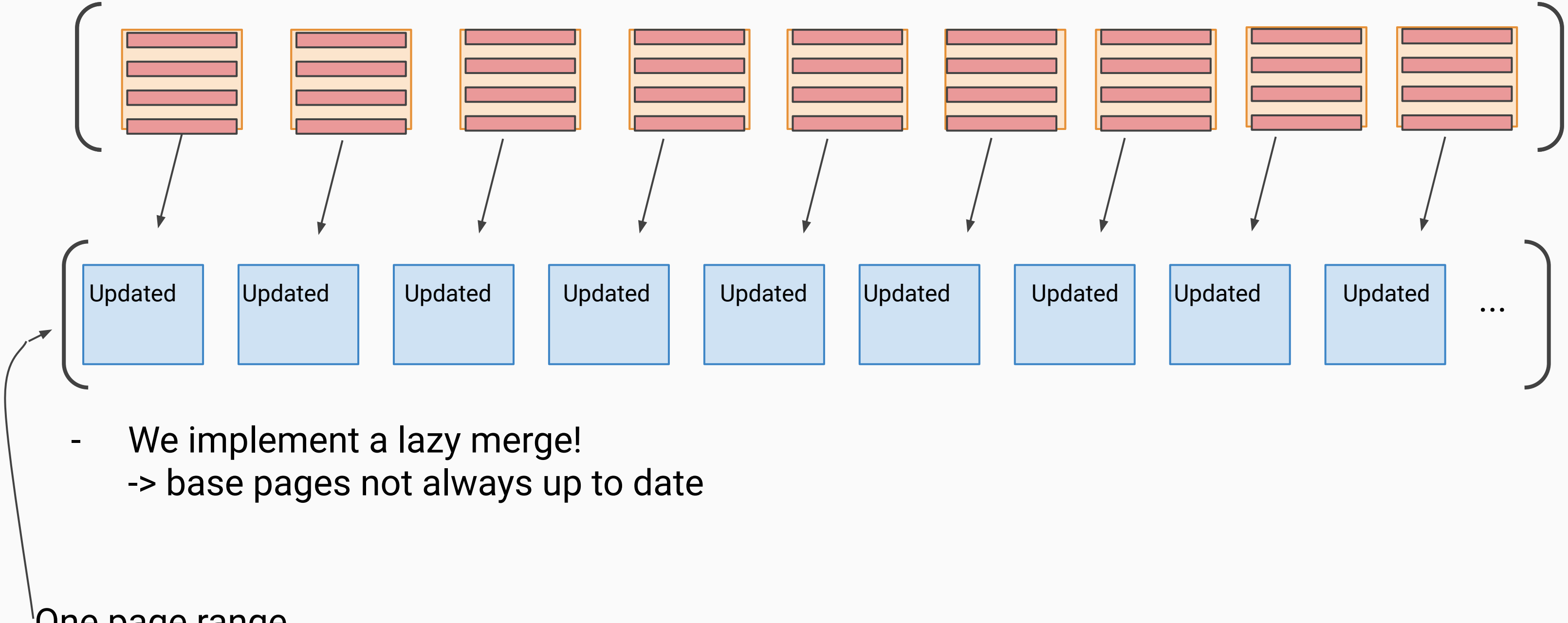


fe d4 0a 83 51 95
7e 82 cc da fc 2c
de b9 14 98
67 b9 71 37 de

Contention Free Merge

Data Reorg: Contention-free Merge Overview

- What is Contention-Free Merge?
  -> Won’t interfere with read or write or affect operation of database



When our tail pages are full

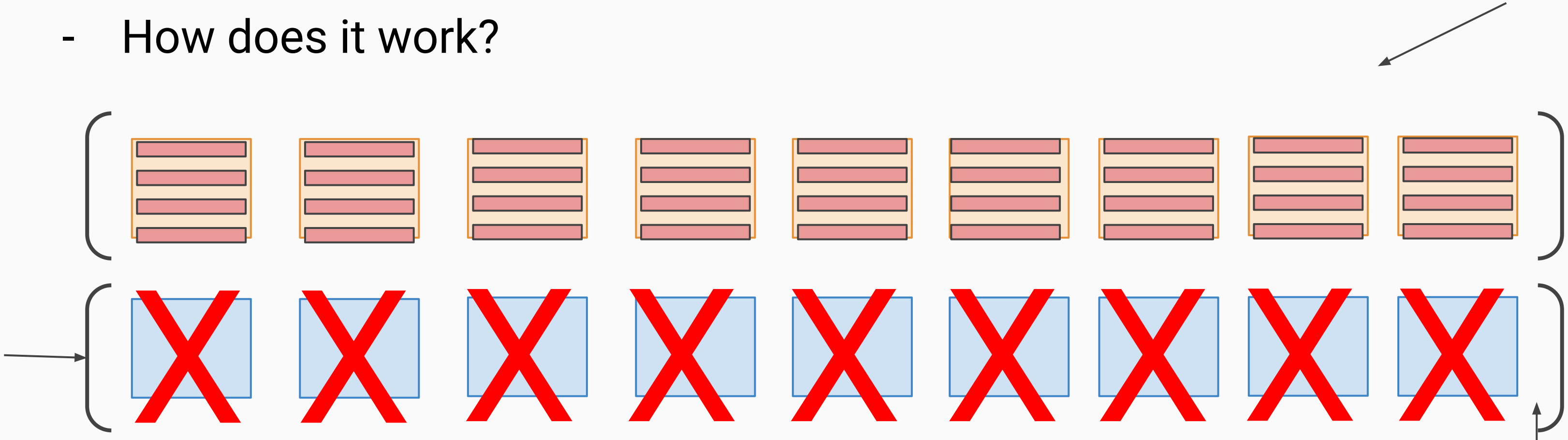
Update the base pages with most recent tail record information

- We implement a lazy merge!
  -> base pages not always up to date

One page range

Data Reorg: Contention-free Merge Implementation

- How does it work?



When tail pages are full

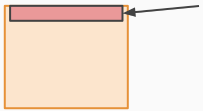
Copy of corresponding basepages

- Iterate backwards through the tail pages and copy the data for every record to its corresponding basepage (for the most up to date tail record)

Insert the newly updated basepages next to the old ones in the same page range

Extras:

- Different thread?
- What is baseRID?
- TPS?



First “record” in every page holds the TPS

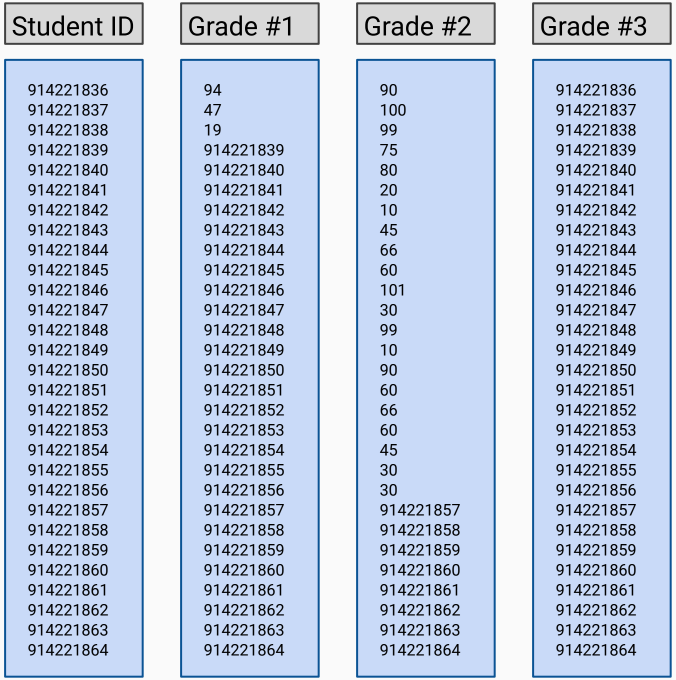
- What happens when update or insert is called while merging?

Indexing

Indexing: Overview

*Examples are not to scale

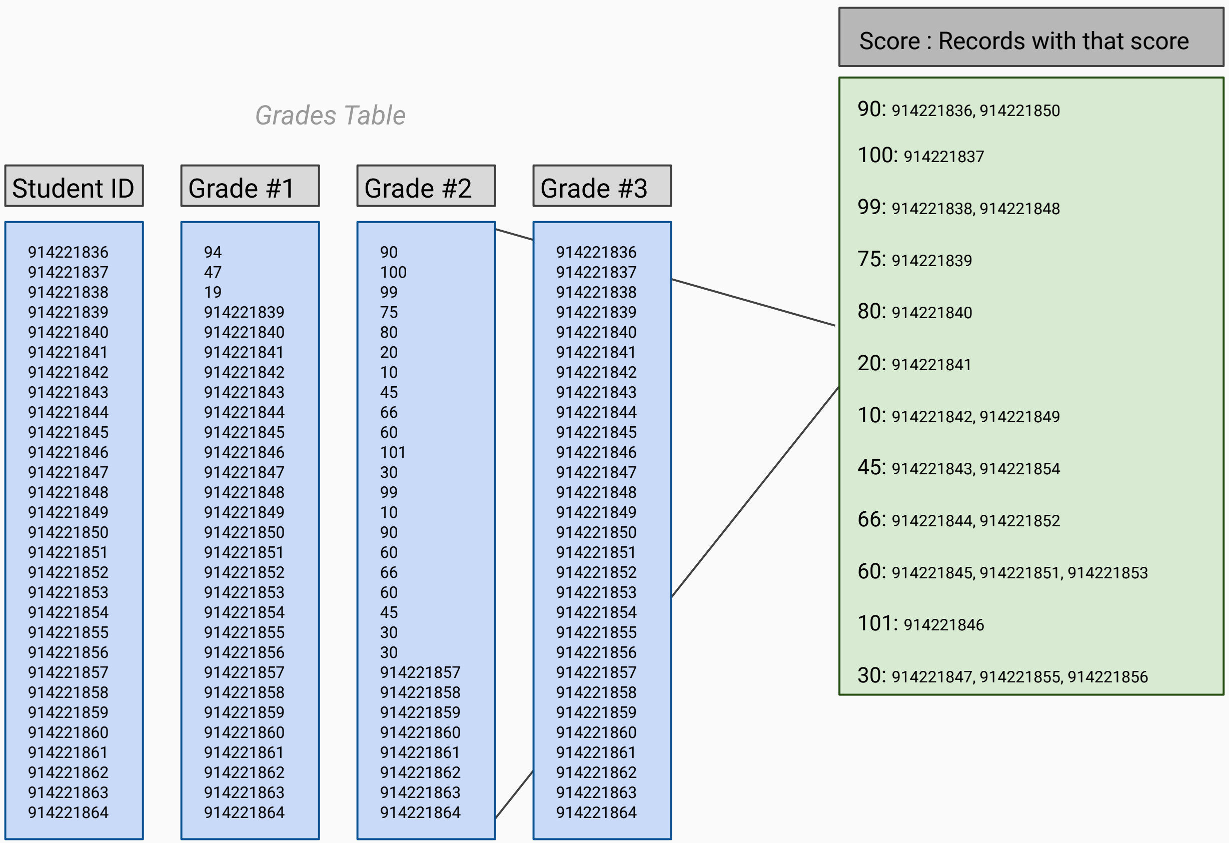
Grades Table



Student ID	Grade #1	Grade #2	Grade #3
914221836	94	90	914221836
914221837	47	100	914221837
914221838	19	99	914221838
914221839	914221839	75	914221839
914221840	914221840	80	914221840
914221841	914221841	20	914221841
914221842	914221842	10	914221842
914221843	914221843	45	914221843
914221844	914221844	66	914221844
914221845	914221845	60	914221845
914221846	914221846	101	914221846
914221847	914221847	30	914221847
914221848	914221848	99	914221848
914221849	914221849	10	914221849
914221850	914221850	90	914221850
914221851	914221851	60	914221851
914221852	914221852	66	914221852
914221853	914221853	60	914221853
914221854	914221854	45	914221854
914221855	914221855	30	914221855
914221856	914221856	30	914221856
914221857	914221857	914221857	914221857
914221858	914221858	914221858	914221858
914221859	914221859	914221859	914221859
914221860	914221860	914221860	914221860
914221861	914221861	914221861	914221861
914221862	914221862	914221862	914221862
914221863	914221863	914221863	914221863
914221864	914221864	914221864	914221864

Indexing: Overview

*Examples are not to scale



Index for Grade #2 Column

Score : Records with that score

90: 914221836, 914221850

100: 914221837

99: 914221838, 914221848

75: 914221839

80: 914221840

20: 914221841

10: 914221842, 914221849

45: 914221843, 914221854

66: 914221844, 914221852

60: 914221845, 914221851, 914221853

101: 914221846

30: 914221847, 914221855, 914221856

Grades Table

Student ID	Grade #1	Grade #2	Grade #3
914221836	94	90	914221836
914221837	47	100	914221837
914221838	19	99	914221838
914221839	914221839	75	914221839
914221840	914221840	80	914221840
914221841	914221841	20	914221841
914221842	914221842	10	914221842
914221843	914221843	45	914221843
914221844	914221844	66	914221844
914221845	914221845	60	914221845
914221846	914221846	101	914221846
914221847	914221847	30	914221847
914221848	914221848	99	914221848
914221849	914221849	10	914221849
914221850	914221850	90	914221850
914221851	914221851	60	914221851
914221852	914221852	66	914221852
914221853	914221853	60	914221853
914221854	914221854	45	914221854
914221855	914221855	30	914221855
914221856	914221856	30	914221856
914221857	914221857	914221857	914221857
914221858	914221858	914221858	914221858
914221859	914221859	914221859	914221859
914221860	914221860	914221860	914221860
914221861	914221861	914221861	914221861
914221862	914221862	914221862	914221862
914221863	914221863	914221863	914221863
914221864	914221864	914221864	914221864

Indexing: Implementation

Results

What We Accomplished

- Broaden our understanding of bufferpool structure
- Implemented database on disk

What We Learned

- Merge algorithm for L-Store
- How to write a database in/out of a binary file

What Is Next

- Milestone 3!

Thank You!