2021-12-21 Besprechungsnotizen

🗓️ Date

21. Dez. 2021

👥 Participants

- @Helen Brüggmann
- @Ragna Solterbeck
- @Felix Köpge
- @Thorsten Papenbrock

🥅 Goals

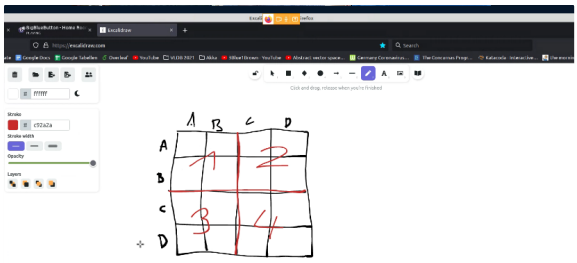
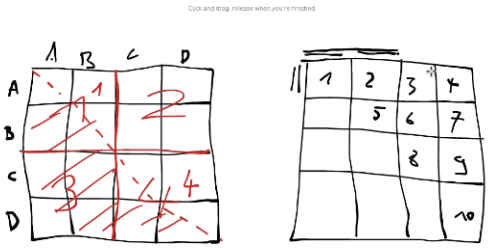
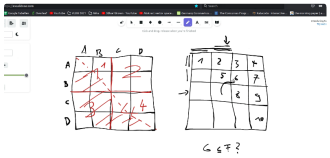
- Projekt Status
- Fragen zum Dynamischen Algorithmus klären
- Datengenerator

🗣️ Discussion topics

Time	Item	Presenter	Notes
	Projektstatus		

	Fragen zum Dynamischen Algorithmus		Welche Algorithmus Ideen gibt es um vom statischen zum dynamischen Algorithmus über zu gehen:
			• Entwickeltes System ist exakt oder eventually-consistent (approximativ)
			  ◦ welche Ziele haben wir?
			  ◦ approximatives System: führt zu long-term Degeneration
			• SQL ist vermutlich zu langsam, eigene Datenstrukturen erfordert
			• Bloom Filter: probabilistische Datenstruktur/"Sketch"
			  ◦ Nur auf 1ster, 2ter, bis zu 3ter Ebene…. nicht überall
			  ◦ Bloom Filter lassen sich schneiden
			  ◦ bei INSERTs exakt, sobald DELETEs durchgeführt werden approximativ
			  ◦ bei vielen DELETE's (prozeduales Limit?) Bloom Filter neu berechnen
			• Sketches, die auf Kardinalitäten basieren
			  ◦ Kandidaten können basiert auf Kardinalitäten ausgeschlossen werden
			  ◦ exakt: bei INSERT hochzählen, bei DELETE runterzählen
			  ◦ min-count-sketch: grobe Schätzung wieviele Werte pro Spalte
			  ◦ contradiction-sketch: anzahl an Widersprüchen zählen, siehe FD-paper
			• Batching
			  ◦ große Datenmengen in Batches aufteilen
			  ◦ in einzelnen Batches kann auf INDs vorgeprüft werden
			  ◦ Windowing (näherungsweise immer nur einen Ausschnitt betrachten)
			  ◦ mitlaufender Bloom-Filter
			• Syntactic Patterns
			  ◦ eine andere Form von Sketches
			  ◦ MIN, MAX, null, datentyp, string-pattern.
			  ◦ int <: float <: string
			  ◦ DELETEs via evidence-trail: Merken, welche Werte zu Änderungen im Pattern geführt haben. Bei Löschen der Werte rückggängig machen. Man kann sich nicht alles merken, d.h.

max-trail-size.

- Caching/LocalDataStore Actor
  - Redundante Übertragungen vermeiden: Infrastruktur anpassen, damit jede Tabelle nur einmal an jedes System übertragen wird
  - Caching Actor läuft pro (Worker-)System, fragt bei InputReader des Master-Systems direkt an
- Vertical Data Partitioning (“Vertikales Schneiden”)
  - Spalten isolieren, auf Systeme verteilen
  - parititionierungs-matrix: partitionierung mit minimaler replizierung
  - räumliche Co-Allokierung auf Spalten-Kombinations-Raum
  - 
  - bei unären INDs muss z.b. unter IND ‘X c Y’ die Spalten X und Y auf dem System co-allokiert sein
  - 
  - gespielte können (B+C, C+B) rausgeschmissen werden
  -
  - 

	Datengenerator		• bestehende Datensätze als Seed-Datensätze nehmen
			  ◦ richtige Daten haben abhänigkeit, die wir bei Random Generation schwer bekommen
			• echtwelt-datensätze werden in papern referenziert => echtwelt-datensätze finden
			  ◦ unendlicher datenstrom = ge-cyclete CSV dateien, einfach wiederholen
			  ◦ zufällige generation = gleich-verteilte abhängigkeiten
			• Randomisiertes Einfügen von Updates, Deletes und Werte-Modifikationen (nach konfigurierbaren Limits, z.b. 5% best Case, 90% worst-case)
			  ◦ Achtung: wenn man Spaltenweise-unabhängig generiert, gehen Abhängigkeiten verloren!
			  ◦ bewusst Widersprüche generieren
			• Dimensions-Tabellen die als Referenz-Tabelle stabil sind; Fakten-Tabellen die wachsen und wachsen => nicht gleichwertig betrachten! macht daten un-realistisch schwer. referenz-tabelle zuerst generieren!

	Hausaufgabe		• soweit gut • n-ary INDs wären noch toll