Lecture 14

COMP207

Overview

- Reminders
- Distributed databases
- Transparencies
- ACID in distributed databases
- Query processing in distributed databases

Tuesday next week = start of week 8 module!

- We are only looking at the module for week 7 today!
- Meaning: Tuesday next week is about the next topic, specifically XML incl. XPath in Module Week 8
- The weekly quiz for in 2 weeks will be based on Week 7 and in 3 weeks Week 8

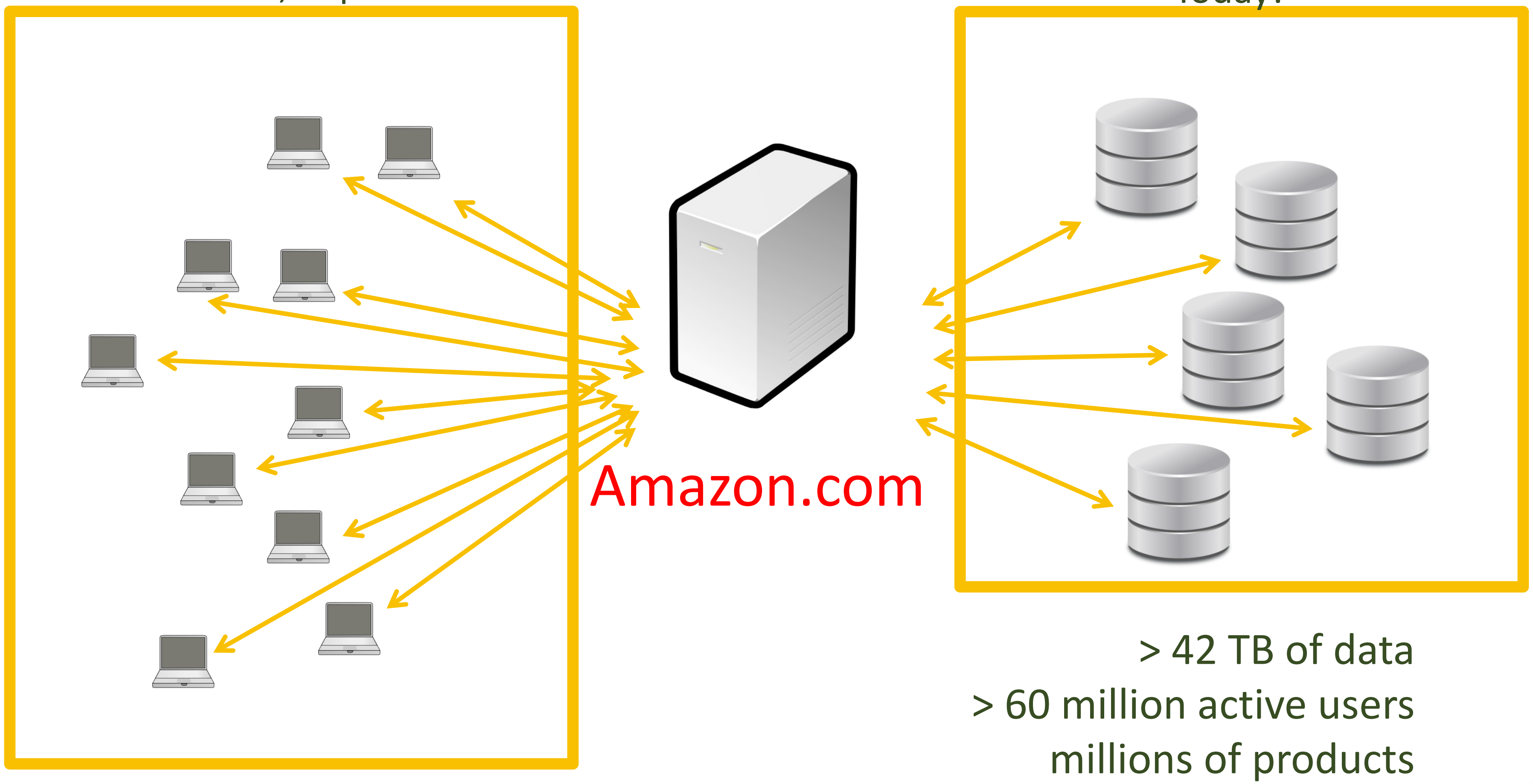
SQL assignment feedback

- Feedback will be released on Tuesday 21st of November
- Even with an exemption from late penalties, you can’t hand-in after Monday the 20th
- You can still submit but unless you have an exemption, you will lose 5 points at every start of a 24-hour period, except
  - Not below 40
  - After 5 days you will lose all points
  - E.g. submitting between yesterday at 17:00:01, to today at 17:00:00 will lose you 5 points, then 10 points between today 17:00:01 to 17:00:00 tomorrow and so on

What is this week all about?

Key problem for ACID, in particular Isolation

Today!



millions of users per day
many at the same time

> 42 TB of data
> 60 million active users
millions of products

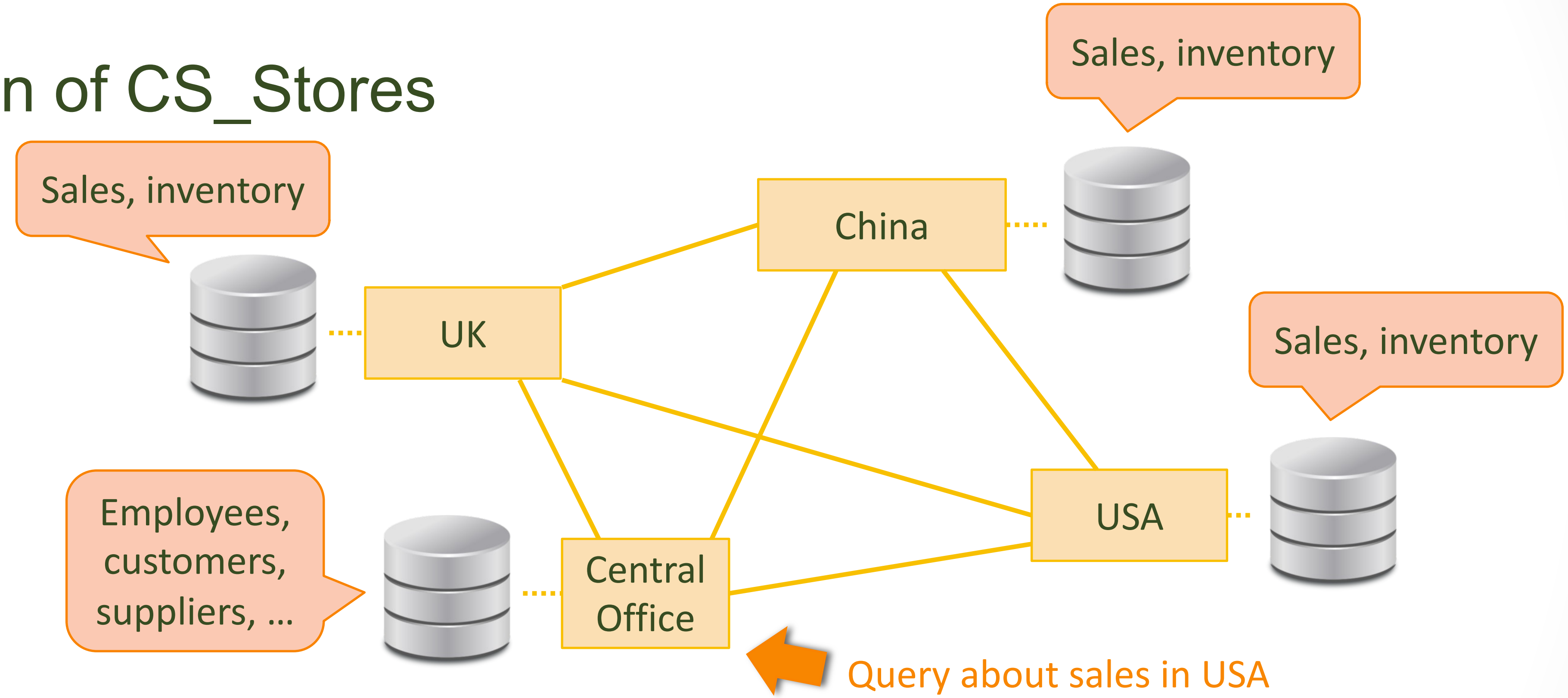
100s of servers

What is the point?

- There are a lot of issues with doing distributed databases, but we can let computers deal with them
- Example: Keep track of transparancies

Fragmentation example

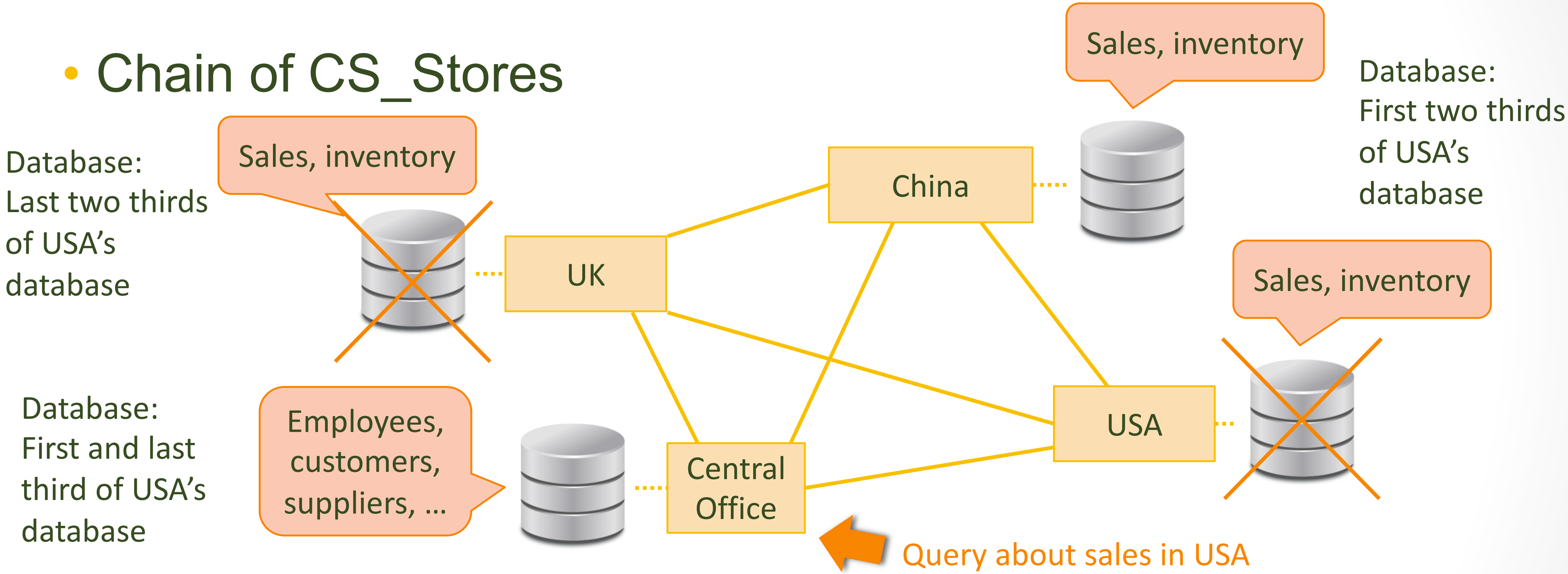
- Chain of CS_Stores



- Each site stores only data primarily relevant to it
- Distributed DBMS provide access to data at all sites

Redudancy example

- Chain of CS_Stores



- Each site stores only data primarily relevant to it AND some additional data to ensure redundancy

Transparencies, what are they good for?

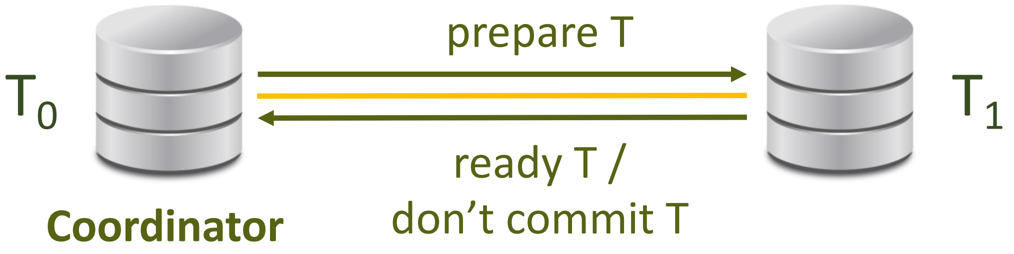
- While it is easy enough to remember simple fragmentation like this, redundancy gets annoying to remember fast
  - Luckily, the point is that we can get the computers to keep track of it for us: We write a simple SQL query, and the computer figures it out for us
- (many other kinds of transparencies, like naming and location)

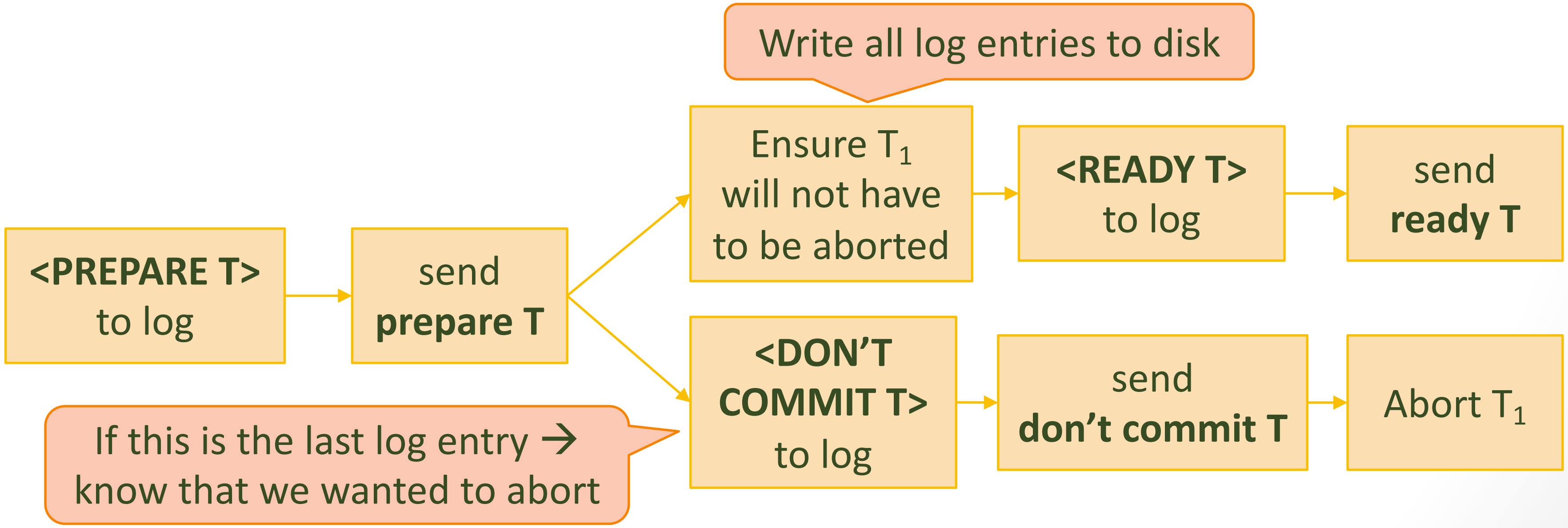
ACID in distributed databases

- Recovery (i.e. atomicity and durability)
  - 2 & 3 phase commit protocol
- Concurrency control (i.e. satisfy consistency and isolation)

Logging: Phase 1

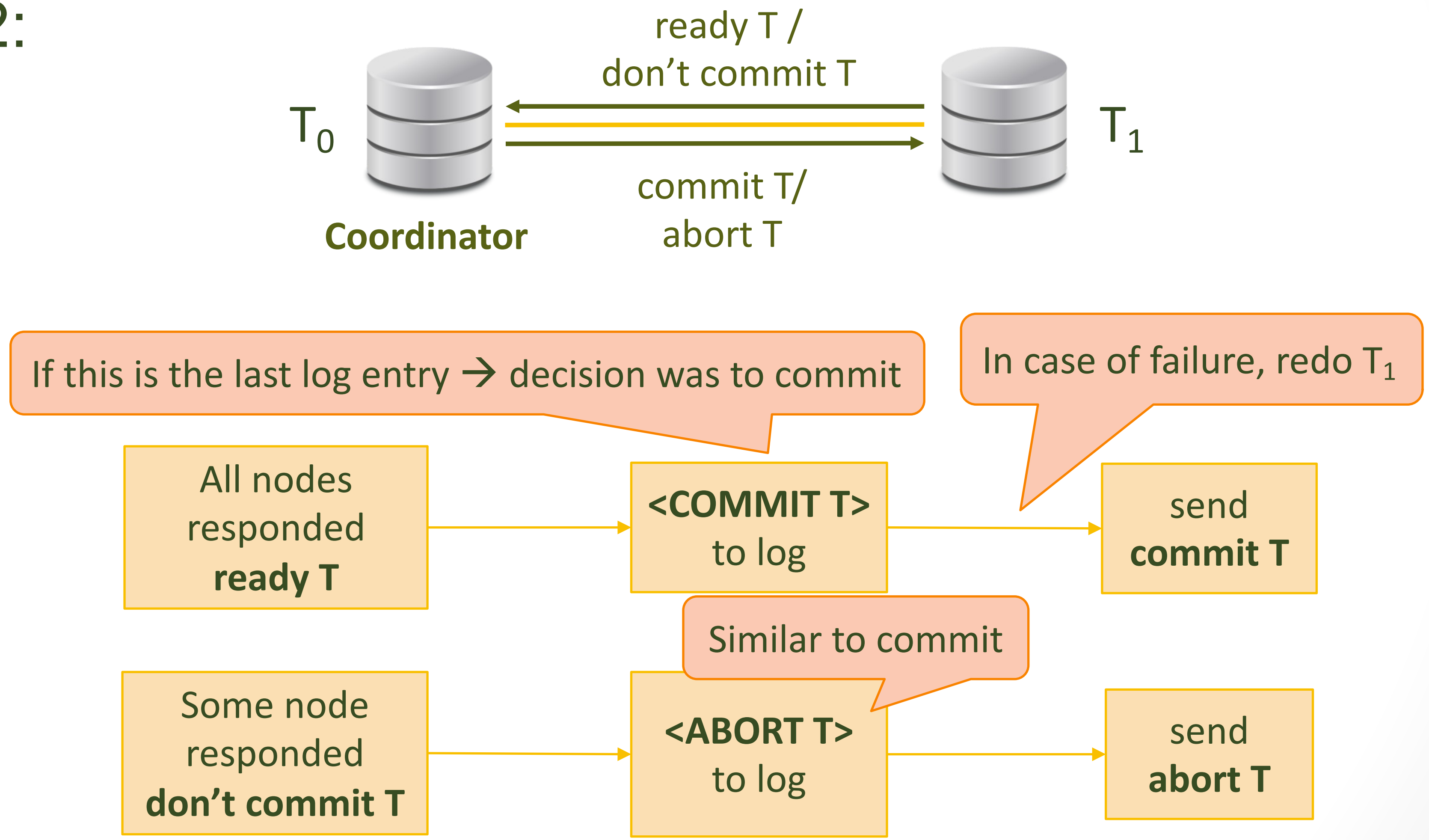
- Again, we have to be careful in which order to write to disk and to the log
- Phase 1:





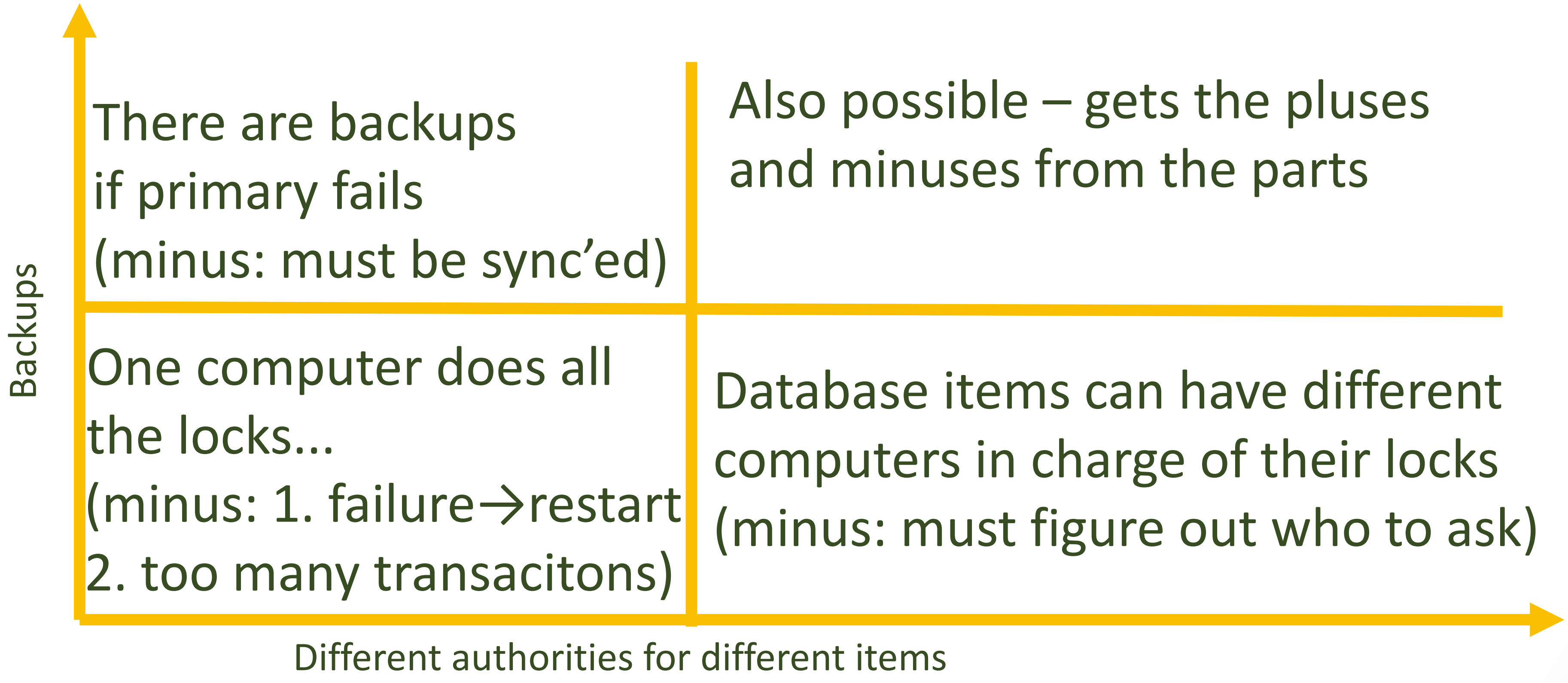
Logging: Phase 2

- Phase 2:

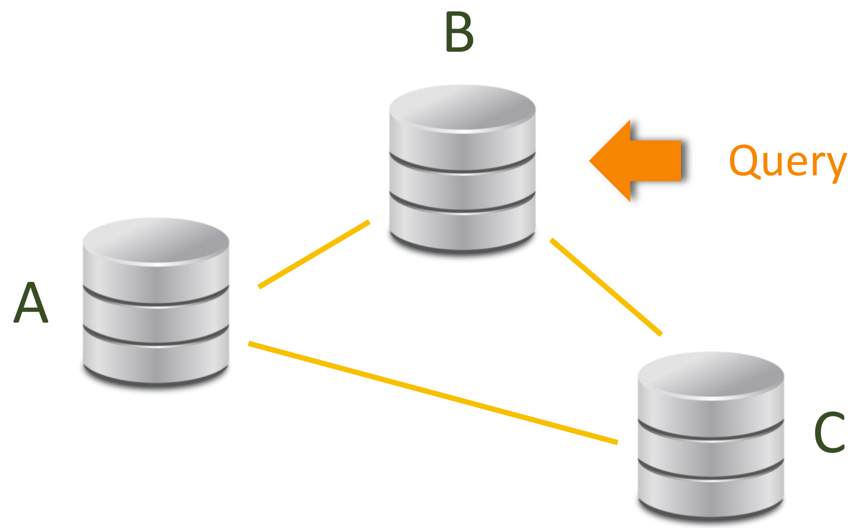


Concurrency control in DDBMS

- For full isolation/consistency, often based on locks:



Query Processing in Distributed DBMS



- Try to answer query at site where query is raised
- If not possible: request information from other sites
  - Slow → design database to reduce this as much as possible
  - Most expensive: joins

Semijoins (⋉)

- $R \ltimes S \;:=\; R \bowtie \pi_{\text{common attributes of R and S}}(S)$

Modules

module	year
COMP105	1
COMP201	2
COMP207	2

Lecturers

name	module
J. Fearnley	COMP105
S. Coope	COMP201

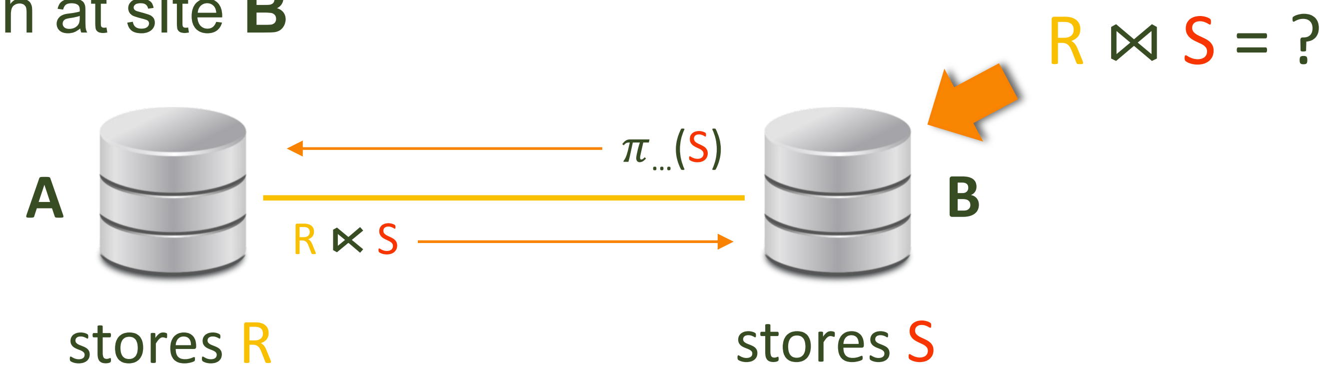
Modules ⋉ Lecturers =

module	year
COMP105	1
COMP201	2

- Intuition: $R \ltimes S$ = set of all tuples in R that NATURAL JOIN
- at least one tuple in S

Semijoin Reduction

- Goal: compute join at site **B**



- With semijoins:
  - Site **B** sends S' := $\pi_{\text{common attributes of R and S}}$(S) to site **A**
  - Site **A** sends R' := R ⋉ S (= R ⋈ S') to site **B**
  - Site **B** outputs R' ⋈ S
- Communication costs ≈

  |S'|×(size of tuple in S') + |R'|×(size tuple in R')