Take-home Midterm: Data at Scale

Problem 1:

A database tracks the assignment of employees to projects. The company is using SCRUM, an agile methodology, so each project has a “Project owner”, which is actually an employee that is responsible for adding tasks in the current SCRUM period and an assigned SCRUM-master, who is another employee that serves as a resource for the team members if they have questions about the process. These roles can (and sometimes should change).

Each employee has a unique employee ID. Employees are assigned to a “home office” as the company has several sites. Sometimes, an employee will be moved to a different home office.

Employees are not always working on only one project. If they do, then they will have a percentage of their time assigned to a project that is not 100%.

Currently, the data is kept in a single table with scheme

(Employee-ID, Employee_First_Name, Employee_Last_Name, Employee_Age, Employee_Home_Office, Employee_Department, Employee_Address, Project_ID, Project_Name, Project_Site, Project_owner, SCRUM-master, Time_percentage)

The Project_owner and SCRUM_master fields contain the Employee-ID of the employees fulfilling these roles.

(1) Identify a potential key for this scheme.
(2) Identify an update anomaly.
(3) Explain why hiring a new employee that is not yet assigned to a project because she needs to still undergo training causes an insert anomaly.
(4) Explain that closing a department (and firing all its employees and deleting all records with employees working in the department) could cause a delete_anomaly.
(5) Propose a better scheme with more than a single table that is in BCNF.
(6) Develop a Neo-style graph database approach for the database.

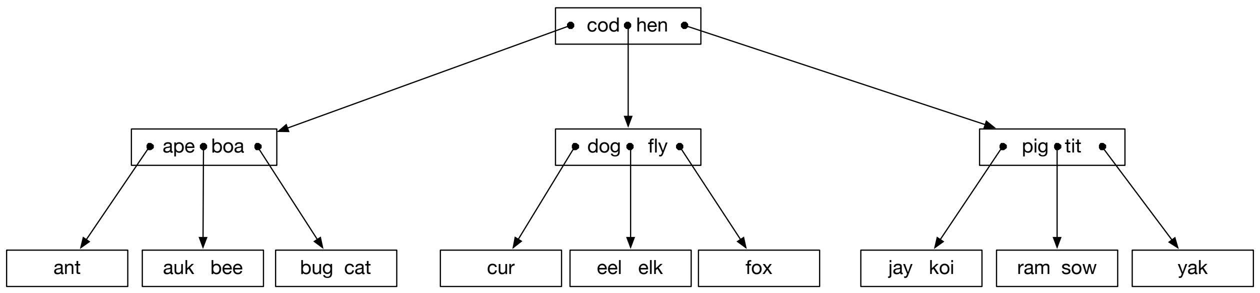
Problem 2:

Give an example that shows that using a locking scheme that locks all records to be read with a read lock before reading and all records affected by a write operation with a write lock, but that is different than Two-Phase Locking (2PL) or strict 2PL does not guarantee serializability.

Problem 3:

A *phantom read* occurs when, in the course of a transaction, new rows are added or removed by another transaction to the records being read. Show by example that locking individual records cannot protect against phantom reads. Instead, one has to use “range locks”.

Problem 4:
Show each result of the successive insertion of “ewe”, “emu”, “dzo”, “rat”, “ray”, “pug” into the following B-tree



Problem 5:

An LH file has 150 buckets. What are the split pointer and the level? Into which buckets do we insert the keys 259 and 390?