(1a) $\pi_{abbreviation, established, name}\ \sigma_{\theta}\ (organization \times is\_member \times country)$

where θ = organization.abbreviation = is_member.organization $\wedge$
is_member.country = country.code $\wedge$ is_member.type = 'member'

(1b)
```
SELECT  name, population
FROM    country
WHERE   code NOT IN (
            SELECT  country1  AS  non_island
            FROM    borders
            UNION
            SELECT  country2  AS  non_island
            FROM    borders )
```

(1c)
```
SELECT    organization,
          SUM (population)  AS  population
FROM      is_member JOIN country
          ON (is_member.country = country.code)
WHERE     type = 'member'
GROUP BY  organization
HAVING    COUNT (code) >= 20
```

(1d) (i) The query returns the scheme (abbreviation) listing the abbreviation of organisations (ordered by the abbreviation) that, for all types of membership that appear in is_member, have at least a country with that type of membership that is part of the organisation.

types: member, observer

result:

abbreviation
CERN

	member	observer
AL		
C	V	
CERN	V	V
CSTO		
EU	V	
NATO	V	
PCA		
WFTU	V	

(ii) $\pi_{abbreviation,type}(organization \overset{\theta}{\bowtie} is\_member) \div \pi_{type} is\_member$

where θ = organization.abbreviation = is_member.organization

(iii)
```
organization_with_all_types(Abbreviation) :-
        organization(Abbreviation, _, _, _),
        ¬organizations_missing_types(Abbreviation).
organizations_missing_types(Abbreviation) :-
        organization(Abbreviation, _, _, _),
      ¬is_member(_, Abbreviation, Type),
        types(Type).
types(Type) :-
      is_member(_, _, Type).
```

(1e)
```
SELECT  abbreviation, established,
        COUNT(CASE WHEN type = 'member' THEN type
                END) AS no_member,
        COUNT(CASE WHEN type = 'observer' THEN type
                END) AS no_observer,
        COUNT(CASE WHEN type NOT IN ('member', 'observer')
                        THEN type
                END) AS no_other
FROM    organization JOIN is_member
        ON (organization.abbreviation = is_member.organization)
GROUP BY abbreviation, established
```

(2c) Order in which locks are acquired:

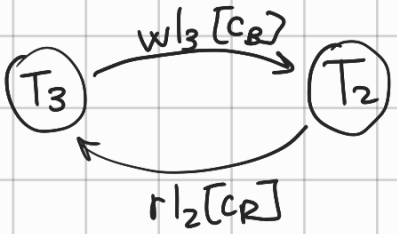
(H1) $rl_1[c_{cz}]$, $wl_1[c_{cz}]$, $rl_1[c_R]$, $wl_1[c_R]$

(H2) $rl_2[c_{cz}]$, $rl_2[c_B]$, $rl_2[c_R]$, $rl_2[c_{GB}]$

(H3) $rl_3[c_R]$, $rw_3[c_R]$, $rl_3[c_B]$, $wl_3[c_B]$

(Highlighted the locks we went to use to create the deadlock)

$r_3[c_R]$, $w_3[c_R]$, $r_2[c_{cz}]$, $r_2[c_B]$, $r_3[c_B]$, <deadlock>



(2b) (i) $S = \{A \to B, A \to D, A \to E, AC \to F, BC \to F, BC \to G, D \to A, G \to F, G \to G, GF \to C\}$

Remove trivial FD $G \to G$.

$BC \to F$ can be obtained by $BC \to G$, $G \to F$, so it is redundant

$GF \to C$ can be simplified to $G \to C$ by using $G \to F$.

Situation so far:

$S = \{A \to B, A \to D, A \to E, AC \to F, BC \to G, D \to A, G \to F, G \to C\}$

Try to remove: $A \to B$

$A^+ = ADE$. $A \to B$ is necessary

Try to remove: $A \to D$

$A^+ = ABE$. $A \to E$ is necessary

E can be obtained only using $A \to E$, so it is a necessary FD.

Try to remove $AC \to F$:

$AC^+ = ACBDEGF$. $AC \to F$ is redundant

$BC \to G$ is necessary because G appears in the RHS of no other FD. Same for $D \to A$. Same for $G \to F$. Same for $G \to C$.

So:

$S_c = \{A \to B, A \to D, A \to E, BC \to G, D \to A, G \to F, G \to C\}$

Either A or E must be in the candidate key.

(ii) AC is a candidate key because $AC^+ = ACBDEGF = R$.

AG is a " " " $G \to C$, so we obtain AC that is known to be a candidate key

DG is a candidate key because using $D \to A$ we obtain AG (candidate key).

Therefore: AC, AG, DG are the candidate keys.

Shouldn't DC also be a candidate key since D -> A?

(iii) $R_1(ABDE)$ $S_{R_1} = \{A \to B, A \to D, A \to E, D \to A\}$

candidate keys: A, B so R_1 is in 3NF

$R_2(BCFG)$ $S_{R_2} = \{BC \to G, G \to F, G \to C\}$

candidate key: BC

R_2 is not is 3NF because of $G \to F$

Decompose $R_2(BCFG)$ using $G \to F$:

$R_3(BCG)$ $R_4(FG)$

R_1, R_3, R_4 is in 3NF

(iv) Since R_1, R_2 are not in 3NF, then they aren't in BCNF.

R_1 is in BCNF since A and D are candidate keys.

R_2 is not in BCNF. Decompose on $G \to F$ (as before)

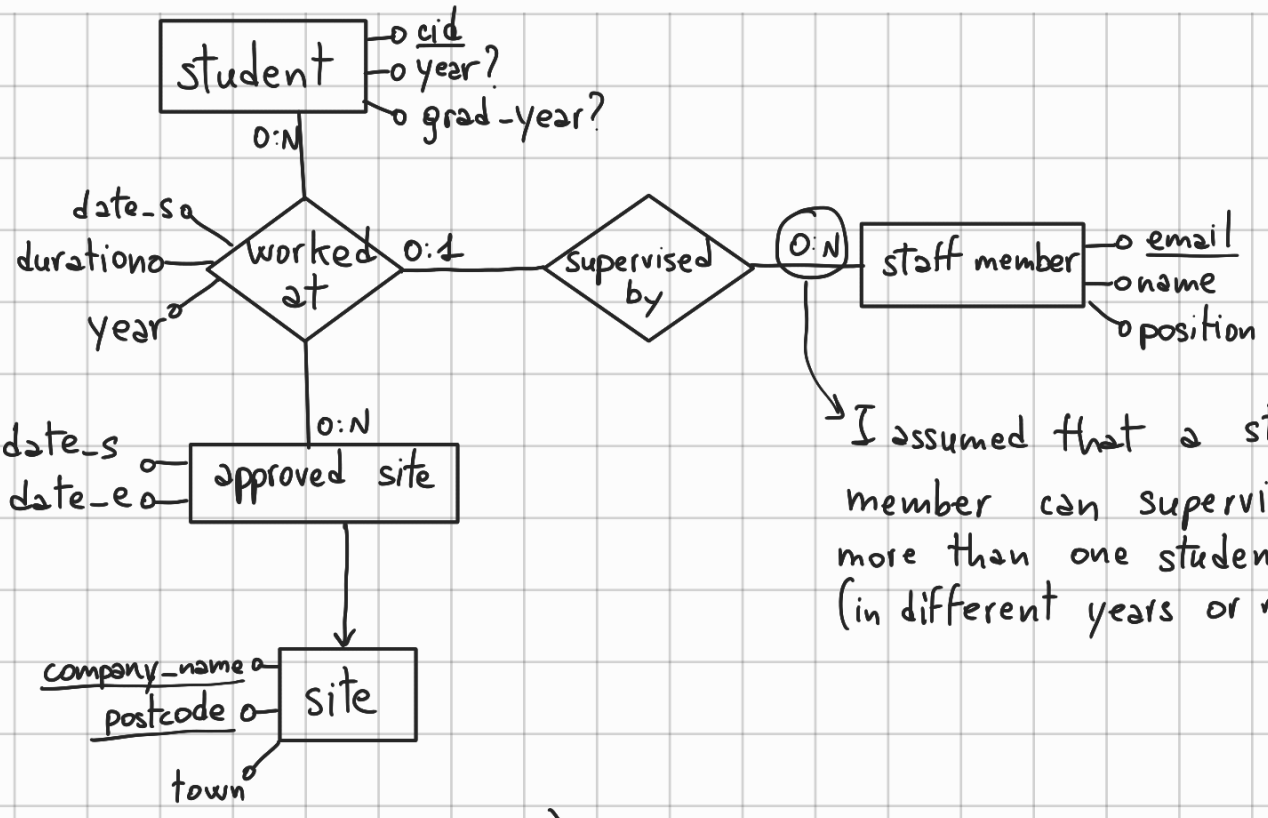
Situation so far: $R_1(ABDE)$, $R_3(BCG)$, $R_4(FG)$

R_3 is not in BCNF. Decompose on $G \to C$:

$R_1(ABDE)$ $R_5(BG)$ $R_6(CG)$ $R_4(FG)$ BCNF

N.B. the projection of S_c into R_5 is $\emptyset$

2a



I assumed that a staff member can supervise more than one students (in different years or not)

student (cid, year?, grad-year?)

site (company_name, postcode, town)

approved_site (company_name, postcode, date_s, date_e)

approved_site (company_name, postcode) $\overset{fk}{\Longrightarrow}$ site (company_name, postcode)

staff_member (email, name, position)

worked_at (cid, company_name, postcode, date_s, duration, year, supervisor_email?)

worked_at (cid) $\overset{fk}{\Longrightarrow}$ student (cid)

worked_at (company_name, postcode) $\overset{fk}{\Longrightarrow}$ approved_site (company_name, postcode)

worked_at (supervisor_email) $\overset{fk}{\Longrightarrow}$ staff_member (email)