# Introduction to Databases

(Typed up in a nice word document so if there are mistakes, please feel free to correct them! Big thanks to Dami, Daniel, and Jacob for contributing answers 😊)

1a) Relational Algebra

πcode country – (πcountry 1 as code borders ∪ πcountry 2 as code borders)

1b) SQL

SELECT name

FROM country

JOIN is\_member

ON is\_member.country = country.code

WHERE is\_member.type <> ‘member’

AND NOT EXISTS (SELECT organisation

FROM is\_member AS turk

WHERE turk.country = ‘TR’

AND turk.organization = is\_member.organization

AND turk.type = ‘member’)

SELECT name

FROM country

WHERE NOT EXISTS (SELECT \*

FROM is\_member

WHERE country.code = is\_membr.country

AND type = ‘member’

AND is\_member.organization IN (SELECT organization

FROM is\_member

WHERE country = ‘TR’

AND type = ‘member’))

1ci) Lists the names and the capitals of countries with a population over 1,000,000 and whose capitals are not the city associated with an organisation

|  |  |
| --- | --- |
| name | capital |
| Switzerland | Bern |
| Turkey | Ankara |

1cii) Relational Algebra

πname, capital σpopulation > 1000000 country –

πname, capital σcountry.code = organization.country ∧ country.capital = organization.city (organization × country)

1ciii) Datalog

non\_organisation\_capitals (Name, Capital) :-

country (Name, Code, Capital, \_, Population),

¬ cities (Capital, Code),

Population > 1000000.

cities (City, Country) :-

organization (\_, City, Country, \_),

isNotNull (City),

isNotNull (Country).

1d) SQL

SELECT DISTINCT name,

length AS longest

FROM country

JOIN borders

ON country.code IN (borders.country1, borders.country2)

WHERE length >= ALL (SELECT length

FROM borders

WHERE country.code IN (borders.country1, borders.country2))

1e) SQL

SELECT name,

continent,

pc\_area

FROM country

JOIN (SELECT code,

100.0 \* area / SUM (area)

OVER (PARTITION BY continent) AS pc\_area

FROM country

JOIN encompasses

ON country.code = encompasses.country

) AS country\_pcs

ON country.code = country\_pcs.code

WHERE pc\_area >= 5

SELECT name,

continent,

pc\_area

FROM (

SELECT country.name,

encompasses.continent,

percentage \* area / SUM(percentage \* area / 100.0)

OVER (PARTITION BY continent) AS pc\_area

FROM country

JOIN encompasses

ON country.code = encompasses.country

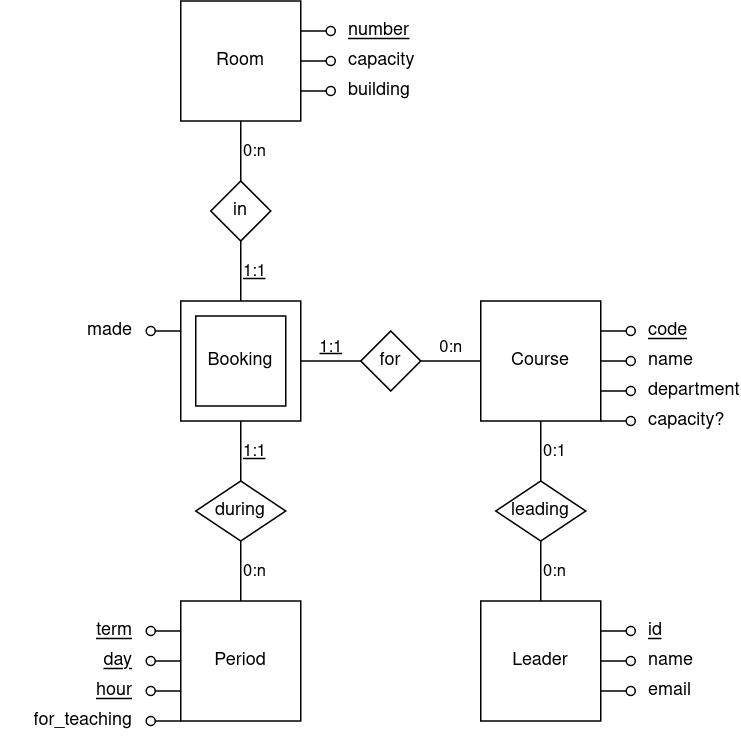
) AS country\_area

WHERE pc\_area >= 5

;

2a)i)

This diagram was made with (<https://app.diagrams.net/>), its .drawio file should be in the same directory as this Word file



2a)ii)

course(code%, name, department, capacity?, leader?)

leader(id%, name, email)

room(number%, capacity, building)

period(term%, day%, hour%, for\_teaching)

booking(in%, term%, day%, hour%, for%, made)

course(leader) => leader(id)

booking(in) => room(number)

booking(term) => period(term)

booking(day) => period(day)

booking(hour) => period(hour)

booking(for) => course(code)

Useful links for Q2:

<http://functionaldependencycalculator.ml/>

<http://www.ict.griffith.edu.au/normalization_tools/normalization/ind.php#editAttibutes>

2bi) S = {AE -> AEF, B -> DC, D -> B, DH -> C, C -> H, F -> E, G -> ACDH}

AE -> AE is trivial

DH -> C, G -> ADH, G -> C => ∅

G -> AD, D -> B, B -> DC, C -> H, G -> H => ∅

D -> B, B -> DC, DH -> C => ∅

Sc = {AE -> F, B -> DC, , C -> H, F -> E, G -> AD}

2bii) G must appear in any candidate key as it cannot be determined by other attributes.

GE and GF are candidate keys as it is impossible to determine E nor F from ABCDGH. Both GE and GF cover the set.

2biii) Only D is shared between Ra and Rb.

D+ = DBCH

As D covers Rb, this is a lossless decomposition. Also, all functional dependencies are preserved as all appear in either Ra or Rb.

2biv) Candidate keys of Ra: GE, GF

Candidate keys of Rb = D, B

AE -> F ✓ (F is prime)

B -> DC ✓ (B is a superkey)

D -> B ✓ (D is a superkey)

C -> H ✕ (Rb is not in 3NF)

F -> E ✕ (Ra is not in 3NF)

G -> AD ✕ (Ra is not in 3NF)

Decomposing Ra on F -> E gives Rc (ADFG) and Rd (EF)

G -> AD still breaks 3NF of Rc

Decomposing Rc on G -> AD gives Re (FG) and Rf (ADG)

Decomposing Rb on C -> H gives Rg (BCD) and Rh (CH)

NlAlternative decomposition: Rd (EF) , Re (FG), Rf (ADG), Rg (BCD), Rh (CH)

R1(EFG) R2(DBC) R3(CH) R4(GAD) R5(AEF)

2ci) Hi = r1[cCZ], r2[cCZ], w1[cCZ], r1[cR], w1[cR], c1, r2[cB], r2[cR], r2[cGB], c2

2cii) H2 and H4 has this proper ty, as there is only one write operation so lost updates and inconsistent analyses are impossible. There is only one possible conflict, on cGB so it has to be equivalent to one of the two possible serial histories (H2H4 or H4H2)

2ciii) Hl = r1[cCZ], w1[cCZ], r1[cR], r3[cR], w1[cR], w3[cR], r3[cR], w3[cB], c1, c3

A lost update occurs in Hl (w1[cR] is ‘lost’), but Hl is recoverable as no transaction i2s committed before another from which it read.