Tuple Relational Calculus

1. Find the names of pizzas that come in a 10 inch size

{T | ∃T1
(T1 ∈ pizza ∧ T1.size = 10
∧ T1.name = T.name)}

2. Find the names of pizzas that come in a 10 inch or a 12 inch size

{T | ∃T1
(T1 ∈ pizza ∧ (T1.size = 10 ∨ T1.size = 12)
∧ T1.name = T.name)}

3. Find the names of pizzas that come in both a 10 inch and a 12 inch size

{T | ∃T1 ∃T2
(T1 ∈ pizza ∧ T2 ∈ pizza ∧ T1.name = T2.name ∧ T1.size = 10 ∧ T2.size = 12
∧ T1.name = T.name)}

4. Find the pairs of different codes of pizzas with the same name and the same size (is there any?)

{T | ∃T1 ∃T2
(T1 ∈ pizza ∧ T2 ∈ pizza ∧ T1.code <> T2.code ∧ T1.name = T2.name ∧ T1.size = T2.size
∧ T.code1 = T1.code ∧ T.code2 = T2.code)}

Yes there some possibly {code} is the key, not {name, size}

5. Find the names and phone numbers of the stores in "College Park" or "Greenbelt" that sell a 10 inch pizza named "pepperoni" for less than $8

{T | ∃T1 ∃T2 ∃T3
(T1 ∈ pizza ∧ T2 ∈ store ∧ T3 ∈ sells ∧ T1.code= T3.code ∧ T2.name = T3.store_name ∧ (T2.area = « College Park » ∨ T2.area = « Greenbelt ») ∧ T1.name = “pepperoni” ∧ T1.size = 10 ∧ T3.price < 8
∧ T2.name = T.name ∧ T2.phone = T.phone)}

6. Find the codes of the most expensive pizzas – assume the scheme of the database is reduced to a relation pizza(<u>code</u>, price) to simplify –

{T | ∃T1 ∀T2
(T1 ∈ pizza ∧ (T2 ∈ pizza ⇒ T1.price ≥ T2.price)
∧ T1.code = T.code)}

7. Find the names of the stores that sell all the pizzas

{T | ∃T1 ∀T2 ∃T3
(T1 ∈ store ∧ (T2 ∈ pizza⇒ (T3 ∈ sells ∧ T2.code= T3.code ∧ T1.name = T3.store_name))
∧ T1.name = T.name)}