CS 3630

Database Design and Implementation

Spring 2021

 **Assignment 4** 

**Goals**

1. Be able to convert between English statements, Relational Algebra expressions, Relational Calculus formulae, and SQL code.

**Note**

You are not allowed to communicate in any form with other classmates on this assignment. A

grade of zero may be given if any communication takes place.

**Introduction:**

The Lego Rebrickable database contains a catalog of Lego parts and their relationship with sets. Sets are a combination of Lego parts that will allow you to create your desired Lego creation. One usage of the Rebrickable database is to identify any missing parts you may have when repurposing one set of Lego parts to construct a creation from another set. More information about the Rebrickable database can be found at <https://rebrickable.com/about/>.



**Database:**

The database that will be used for this assignment is the Lego Rebrickable database shared on the Kaggle website ([https://www.kaggle.com/rtatman/lego-database/data#](https://www.kaggle.com/rtatman/lego-database/data)). Kaggle is a popular data mining and machine-learning competition website. The database contains Lego parts, sets, colors, and inventories from every Lego set in the Rebrickable database from July 2017. I have made some small modifications to this database to make it easier for use.

 **Database Setup**

I have provided my database creation and population script for you as well as the CSV files containing the data. You will only have to import the file using import file wizard. A description of the tables and attributes are provided as comments in the database creation script.



 **ER-Diagram**

 **Useful Symbols**

⟕ ⟖ ⟗ ⨝ σ 𝛑 ≠ ∃ ∧ ∨ 𝞡 𝞮 ¬

σ = Same as “WHERE” Example: σ(customerNumber>200) (customers)

𝛑 = Same as “SELECT” EXAMPLE: 𝛑(cusomerNumber, phone) (customers)

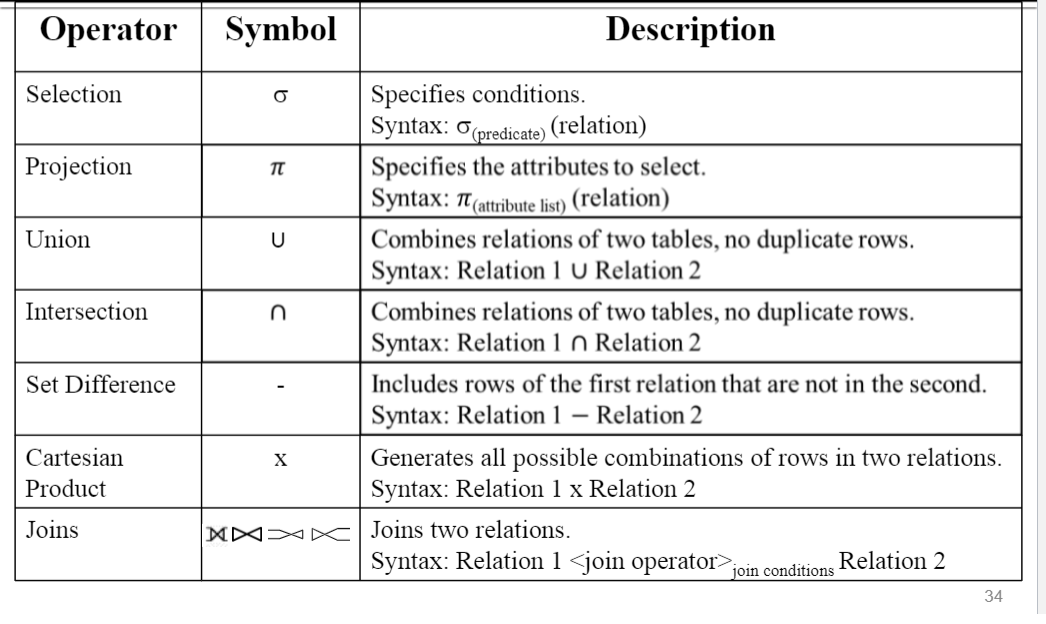
∪ = Same as “Union”

⟕ = Same as “Left outer join”

⟖ = Same as “Right outer join”

⟗ = Same as “Full Outer Join”

⨝ = Same as “Inner Join”



**Generate the SQL code, relational algebra expression, and relational calculus formula for the English statements provided.**

1.

|  |  |
| --- | --- |
| English | Select all the part numbers that are associated with the part category named “Bricks Printed” |
| SQL | SELECT parts.part\_number, part\_categories.category\_name  FROM parts,part\_categories  WHERE part\_categories.category\_name = 'Bricks Printed'  AND parts.part\_category\_id = part\_categories.id; |
| Relational Algebra | 𝛑(parts.part\_number, part\_categories.category\_name)(σ(category\_name = “Bricks Printed”) (part\_categories) AND σ(category\_id, id)(part\_categories, parts)) |
| Relational Calculus | {P | P parts C part\_categories ^ P.part\_number ^ (C.category\_name = “Bricks Printed”) ^ (C.id = P.part\_category\_id)} |

2.

|  |  |
| --- | --- |
| English | Select all part names that belong to the “Minnie Mouse” set. |
| SQL | Select parts.part\_name, sets.set\_name  From parts,sets  Where sets.set\_name = 'Minnie Mouse' AND sets.num\_parts = parts.id; |
| Relational Algebra | 𝛑(part\_number,category\_name)(σ(set\_name = ‘Minnie Mouse’)(sets) AND σ(num\_parts = id)(sets, parts)) |
| Relational Calculus | {P | P parts, S sets ^ (S.set\_name = ‘Minnie Mouse’) ^ (S.num\_parts = P.id) } |

**Generate the English statements, relational algebra expressions, and relational calculus formula for the SQL code provided.**

3.

|  |  |
| --- | --- |
| English | Select the part name and color name from the parts, inner joined with the inventory parts which is inner joined by colors and order the data by part name and color name. |
| SQL | SELECT part\_name, color\_name  FROM parts  INNER JOIN inventory\_parts ON parts.id = inventory\_parts.part\_id  INNER JOIN colors ON inventory\_parts.color\_id = colors.id  ORDER BY part\_name, color\_name; |
| Relational Algebra | 𝛑(part\_name,color\_name)(parts ⨝ parts.id = inventory\_parts.part\_id inventory\_parts ⨝ inventory\_parts.colorid=color.id colors) Order By part\_name, color\_name) |
| Relational Calculus | {P | ∃P parts ∃ C ∈ colors ∃ I ∈ inventory\_parts (parts.id = I.part\_id ^ I.color\_id = C.id))} |

4.

|  |  |
| --- | --- |
| English | Select the set ids as set\_id and theme ids as themes\_id from sets and left outer join with themes if sets theme id is equal to themes id. |
| SQL | SELECT sets.id AS set\_id, themes.id AS themes\_id  FROM sets LEFT OUTER JOIN themes ON sets.theme\_id = themes.id; |
| Relational Algebra | π(sets.id, themes.id) ((⍴sets.id set\_id) ⟕(sets.theme\_id = (⍴themes.id themes\_id) themes (sets)) |
| Relational Calculus | {P | ∃S ∈ sets ∃T ∈ themes (sets.theme\_id = themes.id ^ themes.theme\_name ^ themes.parent\_id) } |

**Generate the English statements, SQL code, and relational calculus formula for the relational algebra expressions provided.**

5.

|  |  |
| --- | --- |
| English | Select the child theme id and parent theme id and name them “themes”. Left outer join child themes with the parent themes whenever the childs parent theme is equal tot the parent theme id. |
| SQL | SELECT child\_theme.id AS themes, parent\_theme.id AS themes  FROM themes LEFT OUTER JOIN child\_theme ON child\_theme.id = parent\_theme.id |
| Relational Algebra | π(child\_theme.id, parent\_theme.id) (  (⍴child\_theme themes) ⟕child\_theme.parent\_id = parent\_theme.id (⍴parent\_theme themes) ) |
| Relational Calculus | {P | ∃T themes ∃C themes (T.id = C.parent\_id ^ T.id = C.id ) |

6.

|  |  |
| --- | --- |
| English | Select the part name from parts where the spare is true, and the inventory parts quantity is greater than 1. Inner join inventory parts with parts id when inventory parts part id is equal to parts id. |
| SQL | Select parts.part\_name  From parts INNER JOIN inventory\_parts ON inventory\_parts.part\_id = parts.id  Where inventory\_parts.is\_spare = true AND inventory\_parts.quantity > 1 |
| Relational Algebra | πpart\_name ((inventory\_parts.is\_spare = TRUE and inventory\_parts.quantity > 1) (  inventory\_parts ⨝inventory\_parts.part\_id = parts.id parts ) |
| Relational Calculus | { P | ∃P ∈ parts ∃ I∈ inventory\_parts (P.id = I.part\_id (I.is\_spare = TRUE ^ I.quantity > 1)(parts,inventory\_parts)} |

**Generate the English statements, SQL code, and relational algebra expression for the relational calculus formula provided.**

7.

|  |  |
| --- | --- |
| English | Select the set id and inventory id for all inventory sets and outer join the inventory sets when the inventory set ids are not equal. |
| SQL | Select inventory\_sets.inventory\_id, inventory\_sets.set\_id  From inventory\_sets OUTER JOIN inventory\_sets ON inventoryset1.set\_id ≠ inventoryset2.set\_id  Where inventory\_sets1.inventory\_id1 = inventory\_sets2.inventory\_id AND inventory\_set1.id = inventory\_set2.id |
| Relational Algebra | π(inventory\_sets.inventory\_id, inventory\_sets.set\_id) (inventory\_sets⟗ inventoryset1.set\_id ≠ inventoryset2.set\_id inventory\_sets ^ σ(inventory\_sets1.inventory\_id1 = inventory\_sets2.inventory\_id AND inventory\_set1.id = inventory\_set2.id) (inventory\_sets)) |
| Relational Calculus | {*P* | ∃IS1 inventory\_sets ∃IS2 inventory\_sets  (IS1.inventory\_id = IS2.inventory\_id ∧ IS1.set\_id ≠ IS2.set\_id ∧ *P*.id = IS1.id)}  2 inventory\_sets with same inventory\_id, but different set id’s |

8.

|  |  |
| --- | --- |
| English | Select the parts, inventory parts, inventory sets, and sets where part id equals the inventory parts part id, the inventory parts inventory id equals the inventory sets inventory id, and sets id equals the inventory sets set id, and the sets’ year < 1990, and the part name of the object is equal to the part name of the parts. |
| SQL | Select parts.id, inventory\_parts.id, inventory\_sets.id, sets.id  From parts,inventory\_parts,inventory\_sets,sets  Where parts.part\_id = inventory\_parts.part\_id AND inventory\_parts.inventory\_id = inventory\_set.inventory\_id AND sets.id = inventory\_sets.set\_id AND year < ‘1990’ AND parts.part\_name = parts.part\_name |
| Relational Algebra | 𝛑(parts.id, inventory\_parts.id, inventory\_sets.id, sets.id)( σ(parts.part\_id = inventory\_parts.part\_id AND inventory\_parts.inventory\_id = inventory\_set.inventory\_id AND sets.id = inventory\_sets.set\_id AND year < ‘1990’ AND parts.part\_name = parts.part\_name) (parts,inventory\_parts,inventory\_sets,sets) |
| Relational Calculus | {*P* | ∃Part parts ∃IP inventory\_parts ∃IS inventory\_sets ∃S sets  (Part.id = IP.part\_id ∧ IP.inventory\_id = IS.inventory\_id ∧ S.id = IS.set\_id ∧  S.year < 1990 ∧ *P*.part\_name = Parts.part\_name)} |

**Submission**

Submit the completed word document on Canvas.

Name this file <your last name>\_assignment\_04.

**Grading**

|  |  |
| --- | --- |
| **Criteria** | **Possible Points** |
| Problem 1 | 12.5 |
| Problem 2 | 12.5 |
| Problem 3 | 12.5 |
| Problem 4 | 12.5 |
| Problem 5 | 12.5 |
| Problem 6 | 12.5 |
| Problem 7 | 12.5 |
| Problem 8 | 12.5 |
|  |  |
| **Total** | 100 |