University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: ___2___

Date: ___2021/10/25_______

Group Number: ___120_______

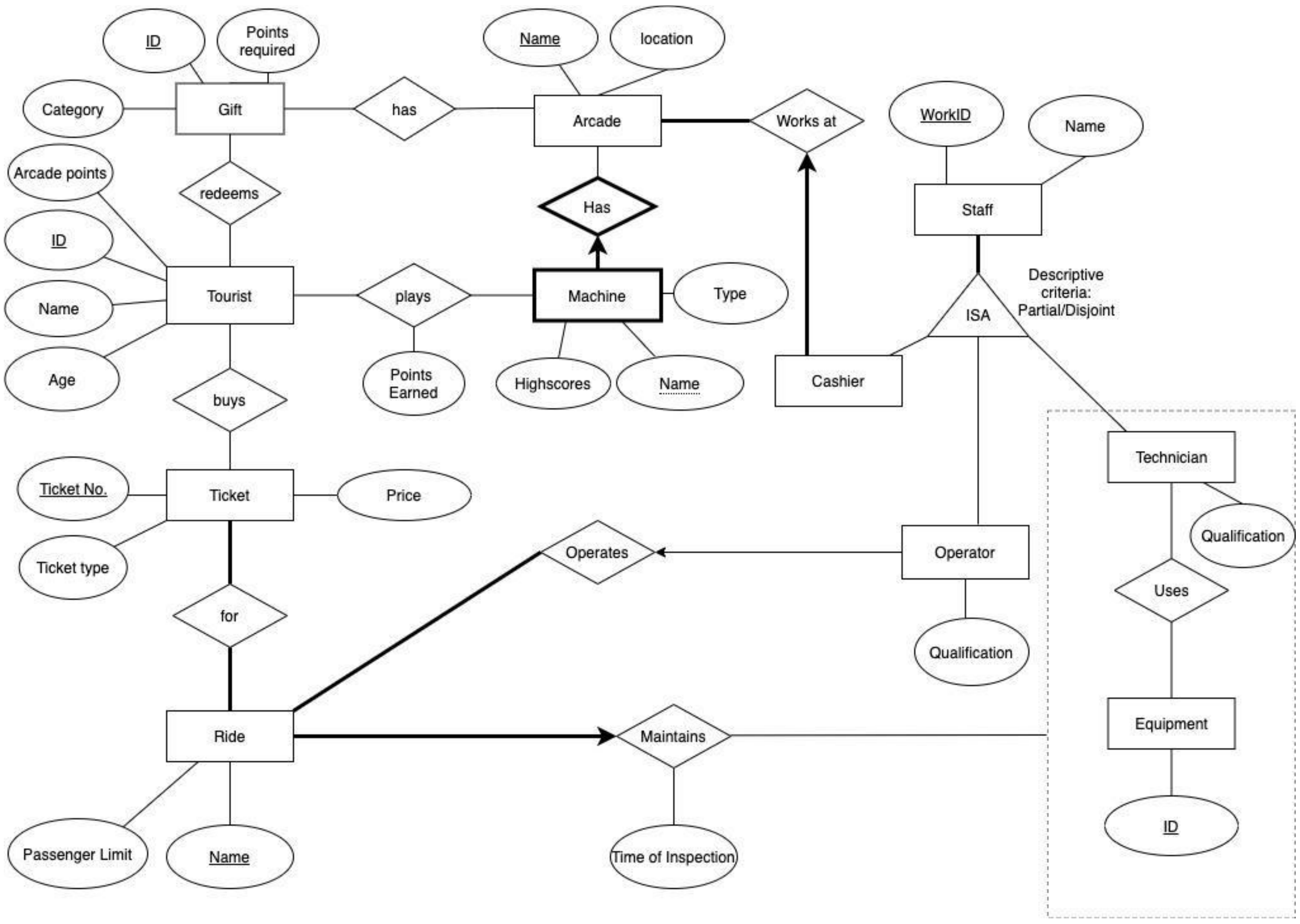
Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Hao Tian (Jack) Gong	44409431	b2z2b	haotiangong@hotmail.com
Weihao Sun	71333785	u6x2b	sunweihao2019@163.com
Yuxiang Fu	94074044	u6y2b	strive2p@student.ubc.ca

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Milestone 2

ER-diagram



In the new ER-diagram, we decided to add a new attribute for the entity ‘gift’ called ‘category’. This attribute consists of stationery, stuffed toys and books etc. We added this attribute to fulfill the FD requirement, and it also fits reality much better. Imagine you earned some points by playing games in an arcade, you may exchange the points for items in desired categories. In our project, we assume that each category is attached with a distinct and unique number of points required. That is, if your points surpass our threshold for a particular category, you are allowed to trade any items within that category.

Schema

Notation: *Underline* means primary key and **Bold** means foreign key.
PK: primary key FK: foreign key

Gift (ID: INTEGER, PointsRequired: INTEGER, Category: CHAR(30))
- Primary Key: ID
- Not null: PointsRequired, Category
- Other constraints: PointsRequired > 0

Tourist (ID: INTEGER, Name: CHAR(30), Age: INTEGER, ArcadePoints: INTEGER)
- Primary Key: ID
- Not null: Name

TouristBuysTicket (**TID**: INTEGER, **TicketNo**: INTEGER)
- Primary Key: (TID, TicketNo)
- Foreign Key: TID references Tourist(ID), TicketNo references Ticket(TicketNo)

Redeems (**GID**: INTEGER, **TID**: INTEGER)
- Primary Key: GID, TID
- Foreign Key: GID references Gift(ID), TID references Tourist(ID)

Ticket (TicketNo: INTEGER, Type: CHAR(30), Price: INTEGER)
- Primary Key: TicketNo
- Not null: Type, Price
- Other constraints: price >= 0

TicketForRide (**TicketNo**: INTEGER, **RideName**: CHAR(50))
- Primary Key: (TicketNo, RideName)
- Foreign Key: TicketNo references Ticket(TicketNo), RideName references Ride_Maintains(RName)

Arcade (Name: CHAR(30), Location: CHAR(50))
- Primary Key: Name
- Unique: Location

ArcadeHasGift (**AName**: CHAR(30), **GID**: INTEGER)
- Primary Key: (AName, GID)
- Foreign Key: AName references Arcade(Name), GID references Gift(ID)

Machine (**<u>AName</u>**: CHAR(30), <u>MName</u>: CHAR(30), Type: CHAR(30), Highscores: INTEGER)

- Primary Key: (AName, MName)
- Foreign Key: AName references Arcade(Name)

TouristPlaysMachine (**<u>TID</u>**: INTEGER, **<u>AName</u>**: CHAR(30), **<u>MName</u>**: CHAR(30), PointsEarned: INTEGER)

- Primary Key: (TID, AName, MName)
- Foreign Key: TID references Tourist(ID), AName references Machine(AName), MName references Machine(MName)

Staff (<u>WorkID</u>: INTEGER, Name: CHAR(30))

- Primary Key: WorkID
- Not null: Name

Cashier_WorksAt (**<u>WorkID</u>**: INTEGER, **AName**: CHAR(30))

- Primary Key: WorkID
- Foreign Key: WorkID, AName
- Not null: AName

Operator_Operates (**<u>WorkID</u>**: INTEGER, **RName**: CHAR(30), Qualification: CHAR(30))

- Primary Key: WorkID
- Foreign Key: WorkID references Staff(WorkID), RName references Ride_Maintains(RName)

Technician (**<u>WorkID</u>**: INTEGER, Qualification: CHAR(30))

- Primary Key: WorkID
- Foreign Key: WorkID references Staff(WorkID)

Equipment (<u>ID</u>: INTEGER)

- Primary Key: ID

Uses (**<u>WID</u>**: INTEGER, **<u>EID</u>**: INTEGER)

- Primary Key: WID, EID
- Foreign Key: WID references Technician(WorkID), EID references Equipment(ID)

Ride_Maintains (<u>RName</u>: CHAR(30), PassengerLimit: INTEGER, **WorkID**: INTEGER, **EID**: INTEGER, TimeofInspection: DATE)

- Primary Key: RName
- Foreign Key: WorkID references Technician(WorkID), EID references Equipment(ID)

- Not null: WorkID, EID, PassengerLimit

Functional Dependencies

Gift (ID, Points Required, Category)
- Category -> PointsRequired
- PK -> everything else

Tourist (ID, Name, Age, ArcadePoints)
- The only functional dependency is the primary key

TouristBuysTicket (**TID**, **TicketNo**)
- The only functional dependency is the primary key

Redeems (**GID**, **TID**)
- The only functional dependency is the primary key

Ticket (TicketNo, Type, Price)
- Type -> Price
- PK -> everything else

TicketForRide (**TicketNo**, **RideName**)
- The only functional dependency is the primary key

Arcade (Name, Location)
- PK -> everything else
- Location (candidate key) -> everything else

ArcadeHasGift (**AName**, **GID**)
- The only functional dependency is the primary key

Machine (**AName**, MName, Type, Highscores)
- The only functional dependency is the primary key

TouristPlaysMachine (**TID**, **AName**, **MName**, PointsEarned)
- The only functional dependency is the primary key

Staff (WorkID, Name)
- The only functional dependency is the primary key

Cashier_WorksAt (<u>**WorkID**</u>, **AName**)

- The only functional dependency is the primary key

Operator_Operates (<u>**WorkID**</u>, **RName**, Qualification)

- Qualification -> RName
- PK -> everything else

Technician (<u>**WorkID**</u>, Qualification)

- The only functional dependency is the primary key

Equipment (<u>ID</u>)

- The only functional dependency is the primary key

Uses (<u>**WID**</u>, <u>**EID**</u>)

- The only functional dependency is the primary key

Ride_Maintains (<u>RName</u>, PassengerLimit, **WorkID**, **EID**, TimeofInspection)

- The only functional dependency is the primary key

Normalization

Decomposition Steps

In the table Gift(ID, PointsRequired, Category), the FD Category -> PointsRequired violates the BCNF rule. Since the closure of Category is {Category, PointsRequired}, Category does not determine all the attributes, so we should decompose this relationship based on this FD. The two relationships after decomposition are Gift_1(ID, Category) and Gift_2(Category, PointsRequired).

Similarly, In the table Ticket(TicketNo, Type, Price), the FD Type -> Price violates the BDCF. The closure of Type is {Type, Price}, so Type cannot determine all the attributes. Hence, we should decompose this relationship to Ticket_1(TicketNo, Type) and Ticket_2(Type, Price).

Also, in the table Operator_Operates(WorkID, RName, Qualification), the FD Qualification -> RName violates the BCNF rule, since the closure of Qualification is {Qualification, RName}. Therefore, Qualification cannot determine all the attributes. We should decompose this relationship to Operator_Operates_1(WorkID, Qualification) and Operator_Operates_2(Qualification, RName).

Table List

Gift_1 (<u>ID:</u> INTEGER, **Category**: CHAR(30))
- Primary Key: ID
- Foreign Key: Category references Gift_2(Category)
- Not null: Category

Gift_2 (<u>Category</u>: CHAR(30), PointsRequired: INTEGER)
- Primary Key: Category
- Not null: PointsRequired
- Other constraints: PointsRequired > 0

Tourist (<u>ID</u>: INTEGER, Name: CHAR(30), Age: INTEGER, ArcadePoints: INTEGER)
- Primary Key: ID
- Not null: Name

TouristBuysTicket (**<u>TID</u>**: INTEGER, **<u>TicketNo</u>**: INTEGER)
- Primary Key: (TID, TicketNo)
- Foreign Key: TID references Tourist(ID), TicketNo references Ticket_1(TicketNo)

Redeems (**<u>GID</u>**: INTEGER, **<u>TID</u>**: INTEGER)
- Primary Key: GID, TID
- Foreign Key: GID references Gift_1(ID), TID references Tourist(ID)

Ticket_1 (<u>TicketNo</u>: INTEGER, **Type**: CHAR(30))
- Primary Key: TicketNo
- Foreign Key: Type references Ticket_2(Type)
- Not null: Type

Ticket_2 (<u>Type</u>: CHAR(30), Price: INTEGER)
- Primary Key: Type
- Not null: Price
- Other constraints: price >= 0

TicketForRide (**<u>TicketNo</u>**: INTEGER, **<u>RideName</u>**: CHAR(50))
- Primary Key: (TicketNo, RideName)

- Foreign Key: TicketNo references Ticket_1(TicketNo), RideName references Ride_Maintains(RName)

Arcade (Name: CHAR(30), Location: CHAR(50))
- Primary Key: Name
- Unique: Location

ArcadeHasGift (**AName**: CHAR(30), **GID**: INTEGER)
- Primary Key: (AName, GID)
- Foreign Key: AName references Arcade(Name), GID references Gift_1(ID)

Machine (**AName**: CHAR(30), MName: CHAR(30), Type: CHAR(30), Highscores: INTEGER)
- Primary Key: (AName, MName)
- Foreign Key: AName references Arcade(Name)

TouristPlaysMachine (**TID**: INTEGER, **AName**: CHAR(30), **MName**: CHAR(30), PointsEarned: INTEGER)
- Primary Key: (TID, AName, MName)
- Foreign Key: TID references Tourist(ID), AName references Machine(AName), MName references Machine(MName)

Staff (WorkID: INTEGER, Name: CHAR(30))
- Primary Key: WorkID
- Not null: Name

Cashier_WorksAt (**WorkID**: INTEGER, **AName**: CHAR(30))
- Primary Key: WorkID
- Foreign Key: WorkID, AName
- Not null: AName

Operator_Operates_1 (**WorkID**: INTEGER, **Qualification**: CHAR(30))
- Primary Key: WorkID
- Foreign Key: WorkID references Staff(WorkID), Qualification references Operator_Operates_2(Qualification)

Operator_Operates_2 (Qualification: CHAR(30), **RName**: CHAR(30))
- Primary Key: Qualification
- Foreign Key: RName references Ride_Maintains(RName)

Technician (<u>**WorkID**</u>: INTEGER, Qualification: CHAR(30))
- Primary Key: WorkID
- Foreign Key: WorkID references Staff(WorkID)

Equipment (<u>ID</u>: INTEGER)
- Primary Key: ID

Uses (<u>**WID**</u>: INTEGER, <u>**EID**</u>: INTEGER)
- Primary Key: WID, EID
- Foreign Key: WID references Technician(WorkID), EID references Equipment(ID)

Ride_Maintains (<u>RName</u>: CHAR(30), PassengerLimit: INTEGER, **WorkID**: INTEGER, **EID**: INTEGER, TimeofInspection: DATE)
- Primary Key: RName
- Foreign Key: WorkID references Technician(WorkID), EID references Equipment(ID)
- Not null: WorkID, EID, PassengerLimit

Creating Tables and Inserting Tuples in SQL

https://github.students.cs.ubc.ca/CPSC304-2021W-T1/project_b2z2b_u6x2b_u6y2b/blob/master/Table_Creation_Insertion.sql

List of Queries

Insertion: Add a tourist to tourist list
Insertion: Add a tourist buying a ticket.
Insertion: Add a gift to the gift list
Insertion: Add a gift redeemed by a tourist.
Insertion: Add an arcade to the arcade list.
Insertion: Add a tourist playing a machine.
Insertion: Add a machine to the machine list.
Insertion: Add a gift in an arcade.
Insertion: Add a cashier working at an arcade.
Insertion: Add an operator operating one ride.
Insertion: Add a technician to the technician list.
Insertion: Add an equipment to the equipment list.
Insertion: Add a technician using equipment.
Insertion: Add a technician using a machine to maintain rides.

Deletion: Delete a tourist buying a ticket.
Deletion: Delete a gift to the gift list
Deletion: Delete a gift redeemed by a tourist.
Deletion: Delete an arcade to the arcade list.
Deletion: Delete a tourist playing a machine.
Deletion: Delete a machine to the machine list.
Deletion: Delete a gift in an arcade.
Deletion: Delete a cashier working at an arcade.
Deletion: Delete an operator operating one ride.
Deletion: Delete a technician to the technician list.
Deletion: Delete an equipment to the equipment list.
Deletion: Delete a technician using equipment.
Deletion: Delete a technician using a machine to maintain rides.

Updating: Haven’t learned yet.
Selection: Haven’t learned yet.
Projection: Haven’t learned yet.
Join: Haven’t learned yet.
Aggregation with Group By: Haven’t learned yet.
Aggregation with Having: Haven’t learned yet.

Nested Aggregation with Group By: Haven’t learned yet.
Division: Haven’t learned yet.