Conceptual Database Design

Entity Sets (Legend: Primary Key, Foreign Key)

Member: M#, FirstName, LastName, Phone#, PName, Balance, Consumption, Tier

Package: PName, C1, C2, StartDate, EndDate, Price

Course: CName, T#, EnrollCount, Capacity, StartDate, EndDate, Day

Trainer: T#, FirstName, LastName, Phone#

Tier: Tier, MinAmount, Discount

Equipment: E#, EType, M#

Transaction: X#, M#, XDate, Amount, XType, Etype

Relationships

Enroll (Member-Package): Cardinality N:1, No Attributes

Consists (Package-Course): Cardinality N:2, No Attributes

Teaches (Trainer-Course): Cardinality 1:N, No Attributes

XActLog (Member-Transaction): Cardinality 1:N, No Attributes

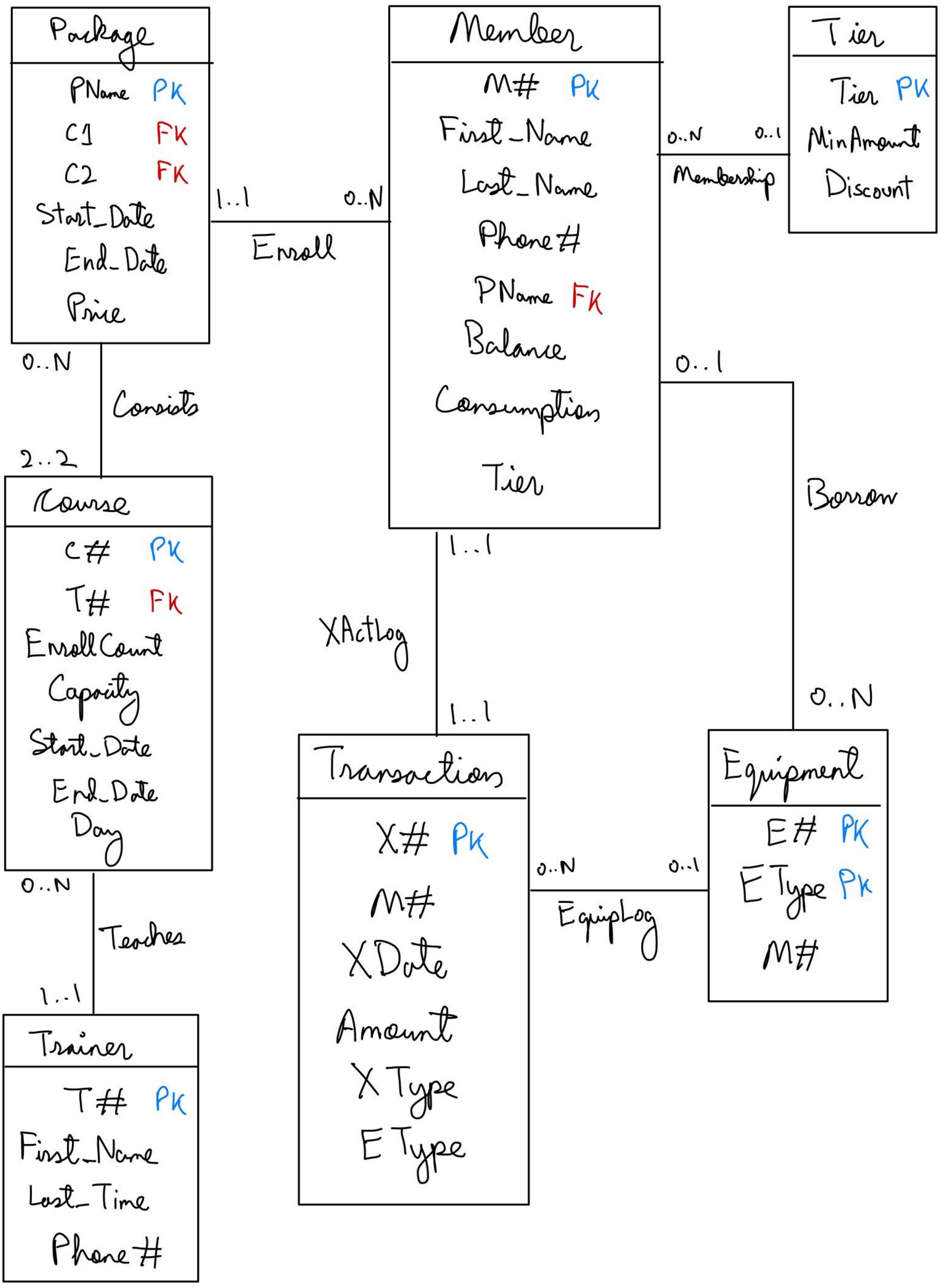
Borrow (Member-Equipment): Cardinality 1:N, No Attributes

EquipLog (Equipment-Transaction): Cardinality 1:N, No Attributes

Membership (Member-Tier): Cardinality N:1, No Attributes

Notes

- StartDate, EndDate, and XDate are all Date objects used to store both date and time. Course Timings are calculated using the times from a Course's StartDate and EndDate.
- XType has 4 valid options:
  - "Credit" for choosing a Package and when registering a new Member. Subtracts the Price of the Package from the Member Balance. Amount is discounted as needed.
  - "Payment" for a payment made by a Member. Adds the Amount entered to the Balance, but applies discounts for Transaction Amount and Member Consumption.
  - "Checkout" for borrowing Equipment of the type EType. Amount becomes the quantity of the Equipment borrowed.
  - "Return" for returning Equipment of the type EType. Amount becomes the quantity of the Equipment returned.
- Equipment has multiple entries for each EType, distinguished by E#. M# is null by default and when returned. M# will be -1 whenever an Equipment becomes lost.
- M# does not have explicit Foreign Key constraints in Transaction and Equipment as the logs must exist even after a Member gets deleted.



Package
PName PK
C1 FK
C2 FK
Start_Date
End_Date
Price
1..1
0..N
Enroll
Member
M# PK
First_Name
Last_Name
Phone#
PName FK
Balance
Consumption
Tier
0..N
0..1
Membership
Tier
Tier PK
MinAmount
Discount
0..N
Consists
2..2
Course
C# PK
T# FK
EnrollCount
Capacity
Start_Date
End_Date
Day
0..N
Teaches
1..1
Trainer
T# PK
First_Name
Last_Time
Phone#
0..1
Borrow
0..N
1..1
XActLog
1..1
Transaction
X# PK
M#
XDate
Amount
X Type
E Type
0..N
0..1
EquipLog
Equipment
E# PK
E Type PK
M#

Logical Database Design (Next 3 Pages)

```
CREATE TABLE Trainer (
   T# INTEGER,
   FirstName VARCHAR2(20) NOT NULL,
   LastName VARCHAR2(20) NOT NULL,
   Phone# VARCHAR2(10) NOT NULL,
   PRIMARY KEY (T#)
);

CREATE TABLE Course (
   CName VARCHAR2(15),
   T# INTEGER NOT NULL,
   EnrollCount INTEGER NOT NULL,
   Capacity INTEGER NOT NULL,
   StartDate DATE NOT NULL,
   EndDate DATE NOT NULL,
   Day VARCHAR2(3) NOT NULL,
   PRIMARY KEY (CName),
   FOREIGN KEY (T#) REFERENCES Trainer,
   CHECK (EndDate > StartDate)
);

CREATE TABLE Package (
   PName VARCHAR2(30),
   C1 VARCHAR2(15),
   C2 VARCHAR2(15),
   StartDate DATE,
   EndDate DATE,
   Price FLOAT NOT NULL,
   PRIMARY KEY (PName),
   FOREIGN KEY (C1) REFERENCES Course,
   FOREIGN KEY (C2) REFERENCES Course,
   CHECK (EndDate > StartDate)
);

CREATE TABLE Tier (
   Tier VARCHAR2(10),
   MinAmount FLOAT NOT NULL,
   Discount FLOAT NOT NULL,
   PRIMARY KEY (Tier)
);

CREATE TABLE Member (
   M# INTEGER,
   FirstName VARCHAR2(20) NOT NULL,
   LastName VARCHAR2(20) NOT NULL,
   Phone# VARCHAR2(10) NOT NULL,
   PName VARCHAR2(20) NOT NULL,
   Balance FLOAT NOT NULL,
   Consumption FLOAT NOT NULL,
   Tier VARCHAR2(10),
   PRIMARY KEY (M#),
   FOREIGN KEY (PName) REFERENCES Package,
   FOREIGN KEY (Tier) REFERENCES Tier
);

CREATE TABLE Equipment (
   E# INTEGER,
   EType VARCHAR2(20),
   M# INTEGER,
   PRIMARY KEY (E#, EType)
);
```

```
CREATE TABLE Transaction (
    X# INTEGER,
    M# INTEGER NOT NULL,
    XDate DATE NOT NULL,
    Amount FLOAT NOT NULL,
    XType VARCHAR2(10) NOT NULL,
    EType VARCHAR2(20),
    PRIMARY KEY (X#)
);
```

Design Decisions:

- Opted to define most fields to be not null to emphasize requirements.
- Package Table has exceptions as we allow the existence of an "empty" Package. This means that since you are permitted to delete courses from a package when updating it, you should conversely be allowed to create a Package with no courses too. As a consequence, such a Package shall not have a StartDate or EndDate field either i.e. set to null.
- The StartDate of a Package is the earlier StartDate between C1 and C2, and similarly EndDate of a Package is the latter StartDate between C1 and C2. They are calculated during Package insertion and updation automatically in Java.
- Member Table has an exception where we allow the existence of a Tier field to be null by default, but it will be updated once they reach certain milestones from the Tier table. It is not an explicit Foreign Key to avoid deletion issues, and the only reason it is part of the table is because the project specifications require the Member to be aware of it.
- Equipment Table has an exception where we allow the existence of the M# field to be null as a default condition for an Equipment entry to not currently be borrowed by any Member.
- Transaction Table has an exception where we allow the existence of an EType field to be null as the table is used for both Package-related transactions as well as checking-out and returning Equipment. It is used only in conjunction with Equipment.
- There are constraints in place to ensure that StartDate is always before the EndDate (even if the date is the same, the timestamps must be different).
- The Amount field of Transaction is unique in the way it is utilized for both displaying the amount of money for a Member's Credit Transaction and Payment Transaction, but also for storing the amount of Equipment of a certain EType that was Checkedout or Returned.
- The remaining constraints are displayed through the schema and ER Model and are self-explanatory. They are also put into effect in the Java DBMS itself through a series of checks for every task that it accomplishes.

Please note that all these schemas are also present in TableGen.java which was used to regenerate tables quickly for testing purposes.

Normalization Analysis (Next 7 pages)

TRAINER TABLE

T#	FirstName	LastName	Phone#

***** FDs *****
{T#} -> {FirstName, LastName, Phone#}

***** CLOSURE *****
{T#} -> {T#, FirstName, LastName, Phone#}
{T#} -> {T#, FirstName, LastName}
{T#} -> {T#, FirstName, Phone#}
{T#} -> {T#, LastName, Phone#}
{T#} -> {FirstName, LastName, Phone#}
{T#} -> {T#, FirstName}
{T#} -> {T#, LastName}
{T#} -> {T#, Phone#}
{T#} -> {FirstName, LastName}
{T#} -> {FirstName, Phone#}
{T#} -> {LastName, Phone#}
{T#} -> {T#}
{T#} -> {FirstName}
{T#} -> {LastName}
{T#} -> {Phone#}

***** MINIMAL COVER *****
{T#} -> {FirstName}
{T#} -> {LastName}
{T#} -> {Phone#}

***** NORMALIZED *****
1st - No set-valued attributes
2nd - All non-prime attributes are FFD on every CK (T#)
3rd - For every non-trivial FD X->A that holds in Trainer, X is a superkey of Trainer.
BC - For every non-trivial FD X->A that holds in Trainer, X is a superkey of Trainer.

<u>COURSE TABLE</u>

<u>CName</u>	**T#**	**Capacity**	**EnrollCount**	**StartDate**	**EndDate**	**Day**

***** FDs *****
{CName} -> {T#, Capacity, EnrollCount, StartDate, EndDate, Day}

***** CLOSURE *****
{CName} -> {T#, Capacity, EnrollCount, StartDate, EndDate, Day}
...
{CName} -> {CName}
{CName} -> {T#}
{CName} -> {Capacity}
{CName} -> {EnrollCount}
{CName} -> {StartDate}
{CName} -> {EndDate}
{CName} -> {Day}

***** MINIMAL COVER *****
{CName} -> {T#}
{CName} -> {Capacity}
{CName} -> {EnrollCount}
{CName} -> {StartDate}
{CName} -> {EndDate}
{CName} -> {Day}

***** NORMALIZED *****
1st - No set-valued attributes
2nd - All non-prime attributes are FFD on every CK (CName)
3rd - For every non-trivial FD X->A that holds in Course, X is a superkey of Course.
BC - For every non-trivial FD X->A that holds in Course, X is a superkey of Course.

PACKAGE TABLE

PName	C1	C2	StartDate	EndDate	Price

***** FDs *****
{PName} -> {C1, C2, StartDate, EndDate, Price}

***** CLOSURE *****
{PName} -> {PName, C1, C2, StartDate, EndDate, Price}
...
{PName} -> {PName}
{PName} -> {C1}
{PName} -> {C2}
{PName} -> {StartDate}
{PName} -> {EndDate}
{PName} -> {Price}

***** MINIMAL COVER *****
{PName} -> {C1}
{PName} -> {C2}
{PName} -> {StartDate}
{PName} -> {EndDate}
{PName} -> {Price}

***** NORMALIZED *****
1st - No set-valued attributes
2nd - All non-prime attributes are FFD on every CK (PName)
3rd - For every non-trivial FD X->A that holds in Package, X is a superkey of Package.
BC - For every non-trivial FD X->A that holds in Package, X is a superkey of Package.

TIER TABLE

<u>Tier</u>	MinAmount	Discount

***** FDs *****
{Tier} -> {MinAmount, Discount}

***** CLOSURE *****
{Tier} -> {Tier, MinAmount, Discount}
...
{Tier} -> {Tier}
{Tier} -> {MinAmount}
{Tier} -> {Discount}

***** MINIMAL COVER *****
{Tier} -> {MinAmount}
{Tier} -> {Discount}

***** NORMALIZED *****
1st - No set-valued attributes
2nd - All non-prime attributes are FFD on every CK (Tier)
3rd - For every non-trivial FD X->A that holds in Tier, X is a superkey of Tier.
BC - For every non-trivial FD X->A that holds in Tier, X is a superkey of Tier.

MEMBER TABLE

M#	FirstName	LastName	Phone#	PName	Balance	Consumption	Tier

***** FDs *****
{M#} -> {FirstName, LastName, Phone#, PName, Balance, Consumption, Tier}

***** CLOSURE *****
{M#} -> {M#, FirstName, LastName, Phone#, PName, Balance, Consumption, Tier}
...
{M#} -> {M#}
{M#} -> {FirstName}
{M#} -> {LastName}
{M#} -> {Phone#}
{M#} -> {PName}
{M#} -> {Balance}
{M#} -> {Consumption}
{M#} -> {Tier}

***** MINIMAL COVER *****
{M#} -> {FirstName}
{M#} -> {LastName}
{M#} -> {Phone#}
{M#} -> {PName}
{M#} -> {Balance}
{M#} -> {Consumption}
{M#} -> {Tier}

***** NORMALIZED *****
1st - No set-valued attributes
2nd - All non-prime attributes are FFD on every CK (M#)
3rd - For every non-trivial FD X->A that holds in Member, X is a superkey of Member.
BC - For every non-trivial FD X->A that holds in Member, X is a superkey of Member.

EQUIPMENT TABLE

E#	**EType**	**M#**

***** FDs *****
{E#, EType} -> {M#}

***** CLOSURE *****
{E#, EType} -> {E#, EType, M#}
{E#, EType} -> {E#, EType}
{E#, EType} -> {EType, M#}
{E#, EType} -> {E#}
{E#, EType} -> {EType}
{E#, EType} -> {M#}

***** MINIMAL COVER *****
{E#, EType} -> {M#}

***** NORMALIZED *****
1st - No set-valued attributes
2nd - All non-prime attributes are FFD on every CK (E#, EType).
3rd - For every non-trivial FD X->A that holds in Equipment, X is a superkey of Equipment.
BC - For every non-trivial FD X->A that holds in Equipment, X is a superkey of Equipment.

TRANSACTION TABLE

X#	M#	XDate	Amount	XType	Etype

***** FDs *****
{X#} -> {M#, XDate, Amount, XType, EType}

***** CLOSURE *****
{X#} -> {X#, M#, XDate, Amount, XType, EType}
...
{X#} -> {X#}
{X#} -> {M#}
{X#} -> {XDate}
{X#} -> {Amount}
{X#} -> {XType}
{X#} -> {EType}

***** MINIMAL COVER *****
{X#} -> {M#}
{X#} -> {XDate}
{X#} -> {Amount}
{X#} -> {XType}
{X#} -> {EType}

***** NORMALIZED *****
1st - No set-valued attributes
2nd - All non-prime attributes are FFD on every CK (X#).
3rd - For every non-trivial FD X->A that holds in Transaction, X is a superkey of Transaction.
BC - For every non-trivial FD X->A that holds in Transaction, X is a superkey of Transaction.

Query Analysis

Chosen Query: Provided an equipment type, display the members who borrowed it this year, the amount borrowed, and when they borrowed it.

This query displays all the different types of equipment available to the user, and then they can choose one of them for our dynamic query. The query informs the user about all the transactions related to borrowing that type of equipment throughout the current year, including the amount of equipment borrowed each time.

There were two reasons behind designing this query:

1. Showcasing the capabilities of the Equipment and Transaction Table, which is not prominent in the basic specifications provided, as well as displaying how the three-way relationship between Member, Transaction, and Equipment works.
2. It is a nice way of perhaps identifying how popular each type of equipment was for the current year, if Members regularly borrow a certain type of equipment, as well as how much they borrow it. This can help identify how to reorganize the Equipment to better suit the needs of the Members and stock accordingly.

By doing so, we are able to not only emphasize on the simplicity that a single Transaction relation offers to collaborate with two other relations (Member and Equipment), who have their own relationship, and not be overloaded.

Additionally, this lets us double down on exemplifying how Transaction table is effectively a logbook, Equipment table is a catalog, and how Member relation can easily interact with them and produce informative results as entries.