UNCC-SRMS

[UNCC Sporting Resources Management System]

Project Team #13 -- Project Report

[Final Report]

Table of contents:

Description of Project Requirements

In order to maintain and manage the many sporting resources held by the university this team plans to implement a database system and web-page interface called the *UNCC - Sporting Resources Management System*. The intended users of the system include: students and faculty who will use the sporting resources, staff who monitor/manage the resources, and administrators who have higher privileges than the staff.

Members:

Students and Faculty of the university will be referred to as *internal members* of this system (i.e. do not require registration, are members in the system by default, and account transactions are handled through the university's billing system). *External members* are then defined as users who are not students or faculty of the university and must register (annually) with a credit card to be recognized by the system.

All members in the system can view the availability of the sporting areas (e.g. courts, fields, etc…), sporting equipment (e.g. basketballs, rackets, etc...), and can reserve these resources for a specific and available time slot. Additionally, all users can cancel any reserved resources or modify them given the current availability of the resources at the time of the modification. For external users, (which must pay a fee upon reserving any sporting areas) the cancellation policy is: If 24 hours or more before reservation, full refund, else (less than 24 hours), 70% of the reservation fee is refunded.

As stated previously, there will also be an accounting information for every member in addition to the obvious details like the member's personal details, history of resources reserved, visiting frequency, misuse reports against them, etc…. For internal users, credits/debits are applied to the user's university account. For external users, credits/debits are applied to the credit card used at the time of registration. This accounting information on a member is how charges are applied in the case of any damages to the resources reserved and/or reservation fees in the case of external users. All fees are applied either at the time of reservation or at the time of misuse being reported. In the case that an external user's credit card expires, their account is locked *(Locked: reservation and use of sporting resources is disabled)* until valid card is provided through the interface.

Any misuse of the sporting resources will result in a misuse report filed against them by a staff member. Two or more misuse reports result in the member's account being locked for a defined duration of two weeks. All members can send questions through the web interface pertaining to a particular sport (see staff details below).

The only difference between the membership types (external vs internal) is that external users must pay a predefined fee for the reserving sporting areas, are not registered in the system by default, and debit/credits are through the defined credit card specified at the time of registration.

Sporting Resources and Staff:

The details regarding every sporting resource (sport *type* supported, sporting *area* or *equipment* item) will be stored in the database. Every type of sport supported by the university (e.g. Basketball, Tennis, etc...) will have a staff associated with it who is in charge of handling questions from internal/external members pertaining to that type of sport. This staff member is also in charge of defining global reservation time windows for that sport type. Staff members should be able to view the details of sporting resources reserved in the past, currently, and any future reservations.

Any available staff member can check-in/out of the resources that have been reserved and upon any damage(s)/loss(es) of the sporting resources by a member, the staff member should be able to report it with all the details necessary to capture the misuse situation (removal of these items from the system happens at this time as well if required). Staff members should also have the ability to add new equipment/remove equipment from the system. The contact information of the staff members is also maintained so that it is accessible to other staff members in case it is needed.

The only other role within the database is that of the administrator. The admin for the database has all the privileges to view, access, and modify any kind of details in the database. This becomes necessary when it comes to tasks such as: Adding/removing staff members or any sports induced on campus, modifying what staff member governs a sport, fixing mistakes made by staff members, etc.

Noun and Verb Identification:

Nouns:

Noun:	Keep?	As a:	Represented/Owning Entity:
database system	no	-	-
web-page	no	-	-
categories	no (context)	-	-
user	kind of	Entity	Member
privilege	kind of	Attribute	Member/Staff/Admin
information	no (too vague)	-	-
sporting resource	yes	Entity	Sporting-Resource
student	yes	Attr. Val	Internal-Member
faculty	yes	Attr. Val	
staff	yes	Entity	Staff
administrator	yes	Entity	Admin
member	yes	Entity	Member
reservation	yes	Relation	Member <--> Sporting-Resource
sporting area	yes	Entity	Sporting-Area
sporting equipment	yes	Entity	Sporting-Equipment
internal member	yes	Entity	Internal-Member
external member	yes	Entity	External-Member
transaction	yes	Entity	Transaction
registration	yes	Attribute	External-Member
credit card	yes	Attribute	External-Member
availability	yes	Attribute	Sporting-Resource
time slot	yes	Attribute	Reservation

fee/charge	yes	Attribute	Sporting-Resource/External-Member/Misuse
misuse report	yes	Entity	Misuse-Report
damage	kind of	Entity	Misuse-Report
account lock	yes	Attribute	Member
question/message	yes	Entity	Message
sport type	yes	Entity	Sporting-Category
associated staff	kind of	Relation	Staff <--> Sporting-Category
time window	kind of	Attribute	Sporting-Resource
detail	no (too vague)	-	-
staff contact information	yes	Attribute(s)	Staff

Verbs:

Verb:	**Keep?**	**As a:**	**Represented/Owning Entity:**
(equipment) loss	kind of	Attr. Val.	Misuse-Report
(resource) removal/add	yes	Relation	Staff <--> Sporting-Equipment
manage	kind of	Relation	Staff <--> Sporting-Category
cancel	yes	Function & Attribute	Reservation & [Function (UI & system ability)]
modify	yes	Function	[Function (UI & system ability)]
check-in/check-out	yes	Attribute	Reservation
handle	no (too vague)	-	-
register	yes	Attribute	External-Member
recognize	no	-	-
view	no	-	-
refund	yes	Attr. Val	Transaction
visit	no	-	-
expire	kind of	Function	[Function (check for expire conditions)]
lock	yes	Entity	Account-Lock
define	no	-	-
pay	kind of	Entity	Transaction
report	kind of	Entity	Misuse-Report
access	no (too vague)	-	-
govern	kind of	Relation	Staff <--> Sporting-Category
fix	kind of	Function	[Function (Admin ability, not logged)]

Entity and Relationship Definitions:

Entity :: Internal-Member [Child of Member Entity]

university_id
type
member_id
password
first_name
last_name
middle_name
address_1
address_2
city
state
zip
phone_num
email
date_of_birth

Entity :: External-Member [Child of Member Entity]

member_id
registration_date
password
first_name
last_name
middle_name
address_1
address_2
city
state
zip
phone_num
email
date_of_birth
card_number
name_of_card_holder
cvv
expiry_date

Entity :: Staff [Child of Member Entity]

role
university_id
type
member_id
password
first_name
last_name
middle_name
address_1
address_2
city
state
zip
phone_num
email
date_of_birth

Entity :: Sporting-Area [Child of Sporting-Resource Entity]

resource_id
category_id *
maximum_reservation_duration
resource_name
description_short
description_long
reservation-cost
date_added

** category_id is a foreign key sporting_category_id From sporting_category*

Table [repair]

resource_id *
date
cost
description

** resource_id is a foreign key resource_id from Sporting_Area*

Entity :: Sporting-Equipment [Child of Sporting-Resource Entity]

equipment_id
category_id *
maximum_reservation_duration
resource_name
description_short
description_long
reservation-cost
date_added
replacement_cost

** category_id is a foreign key sporting_category_id From sporting_category*

Table [Damage]

equipment_id *
date
description

** resource_id is a foreign key resource_id from Sporting_Equipment*

Entity :: Transaction

transaction_id
member_id *
amount
date
type
description

* member_id is a foreign key from the union of staff, internal_member, and external_member tables

Entity :: Account-Lock

member_id *
start_date
end_date
type

** member_id is a foreign key from the union of (member_id) external_member & internal_member tables*

Table :: Message

message_id
sender_id*
reciever_id**
title

* sender_id is a foreign key member_id Member
** reciever_id is a foreign key member_id Member

Entity:: MessageContent

message_id *
content
title
date
read_status

* message_id is a foreign key message_id from Message

Entity :: Sporting-Category

sporting_category_id
name
description
member_id *

* member_id is a foreign key member_id from Staff

Binary Relationship

Member reserves_area Sporting-Resources

member_id *
sporting_area_id **
date
check-in
check-out
scheduled_start
scheduled_end
was_canceled
date_canceled

* member_id is a foreign key member_id from Member
** sporting_area_id is a foreign key area_id from Sporting_area

Binary Relationship
Member reserves_equipment Sporting-equipment

member_id *
sporting_equipment_id **
date
check-in
check-out
scheduled_start
scheduled_end
was_canceled
date_canceled

Binary Relationship
Staff governs Sporting-Category

sporting_category_id **

Binary Relationship
Staff Adds Sporting-Equipment

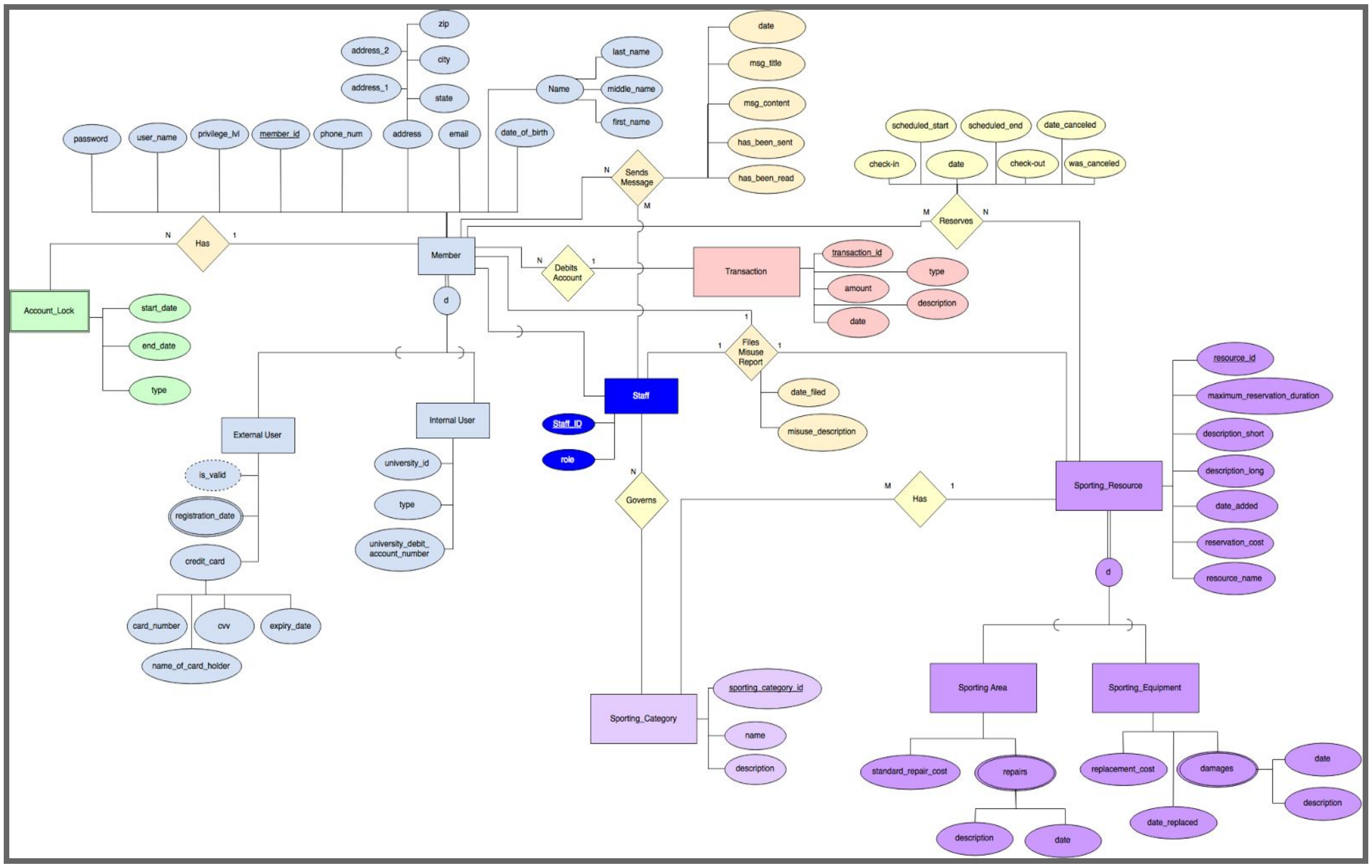
Binary Relationship
Staff Removes Sporting-Equipment

Binary Relationship
Files Misuse-Report On Member, Staff, Sporting-Resource

date

EER Diagram:

(Actual document [possibly updated version] can be viewed with the link: here)



Logical Design:

SportingArea		
resource_id	INT	NOT NULL, UNIQUE
maximum_reservation_duration	INT	NOT NULL
resource_name	VARCHAR(45)	NOT NULL
description_long	VARCHAR(128)	
description_short	VARCHAR(30)	
date_added	DATETIME	
category_id	INT	NOT NULL

Primary Key(s):	resource_id
Foreign Key(s):	category_id references SportingCategory [category_id]
Highest Normal Form:	4NF (at least)
Comments:	

SportingEquipment

resource_id	INT	NOT NULL, UNIQUE
maximum_reservation_duration	INT	NOT NULL
resource_name	VARCHAR(45)	NOT NULL
description_long	VARCHAR(128)	
description_short	VARCHAR(45)	
date_added	DATETIME	
category_id	INT	NOT NULL
replacement_cost	DOUBLE	NOT NULL

Primary Key(s):	resource_id
Foreign Key(s):	category_id references SportingCategory [category_id]
Highest Normal Form:	4NF (at least)
Comments:	

Member

member_id	INT	NOT NULL, UNIQUE
account_type	VARCHAR(10)	NOT NULL
password	VARCHAR(16)	NOT NULL

Primary Key(s):	member_id
Foreign Key(s):	
Highest Normal Form:	4NF (at least)
Comments:	

InternalMember

university_id	INT	NOT NULL
type	VARCHAR(16)	NOT NULL
first_name	VARCHAR(45)	NOT NULL
last_name	VARCHAR(45)	NOT NULL
middle_name	VARCHAR(45)	
address_1	VARCHAR(45)	NOT NULL
address_2	VARCHAR(45)	
city	VARCHAR(45)	NOT NULL
state	VARCHAR(2)	NOT NULL
zip_code	INT	NOT NULL
phone_num	VARCHAR(16)	NOT NULL
date_of_birth	DATE	NOT NULL
email	VARCHAR(45)	NOT NULL

Primary Key(s):	university_id
Foreign Key(s):	university_id references Member [member_id]
Highest Normal Form:	2NF
Comments:	This table violates 3NF because it has a transitive functional dependency namely zip_code → {city state} among non-key attributes. It has been kept as it is because it is assumed that users will mostly wish to display member details including the full address. If the table is normalized to move city state into a separate table along with zip_code it will require a join each time a member's full address needs to be retrieved.

ExternalMember

member_id	INT	NOT NULL
type	VARCHAR(16)	NOT NULL
first_name	VARCHAR(45)	NOT NULL
last_name	VARCHAR(45)	NOT NULL
middle_name	VARCHAR(45)	
address_1	VARCHAR(45)	NOT NULL
address_2	VARCHAR(45)	
city	VARCHAR(45)	NOT NULL
state	VARCHAR(2)	NOT NULL
zip_code	INT	NOT NULL
phone_num	VARCHAR(16)	NOT NULL
date_of_birth	DATE	NOT NULL
registration_date	DATETIME	NOT NULL
card_number	VARCHAR(16)	NOT NULL
name_of_card_holder	VARCHAR(45)	NOT NULL
cvv	INT	NOT NULL
expiry_date	VARCHAR(4)	NOT NULL
email	VARCHAR(45)	NOT NULL

Primary Key(s):	member_id
Foreign Key(s):	member_id references Member [member_id]
Highest Normal Form:	2NF
Comments:	This table violates 3NF because it has a transitive functional dependency namely zip_code → {city state} among non-key attributes. It has been kept as it is because it is assumed that users will mostly wish to display customer details including the full address. If the table is normalized to move city state into a separate table along with zip_code it will require a join each time a customer's full address needs to be retrieved.

Staff

university_id	INT	NOT NULL
role	VARCHAR(10)	NOT NULL
first_name	VARCHAR(45)	NOT NULL
last_name	VARCHAR(45)	NOT NULL
middle_name	VARCHAR(45)	
address_1	VARCHAR(45)	NOT NULL
address_2	VARCHAR(45)	
city	VARCHAR(45)	NOT NULL
state	VARCHAR(2)	NOT NULL
zip_code	INT	NOT NULL
phone_num	VARCHAR(16)	NOT NULL
date_of_birth	DATE	NOT NULL
email	VARCHAR(45)	NOT NULL

Primary Key(s):	university_id
Foreign Key(s):	university_id references Member [member_id]
Highest Normal Form:	2NF
Comments:	This table violates 3NF because it has a transitive functional dependency namely zip_code → {city state} among non-key attributes. It has been kept as it is because it is assumed that users will mostly wish to display customer details including the full address. If the table is normalized to move city state into a separate table along with zip_code it will require a join each time a customer's full address needs to be retrieved.

AccountLock

member_id	INT	NOT NULL
start_date	DATETIME	NOT NULL
end_date	DATETIME	NOT NULL
type	VARCHAR(16)	NOT NULL

Primary Key(s):	member_id start_date
Foreign Key(s):	member_id references Member [member_id]
Highest Normal Form:	4NF (at least)
Comments:	

Transaction

transaction_id	INT	NOT NULL, UNIQUE
member_id	INT	NOT NULL
amount	DOUBLE	NOT NULL
date	DATETIME	NOT NULL
type	VARCHAR(16)	NOT NULL
description	VARCHAR(45)	

Primary Key(s):	transaction_id
Foreign Key(s):	member_id references Member [member_id]
Highest Normal Form:	4NF
Comments:	

Message

message_id	INT	NOT NULL, UNIQUE
sender_id	INT	NOT NULL
reciever_id	INT	NOT NULL
title	VARCHAR(45)	NOT NULL

Primary Key(s):	message_id
Foreign Key(s):	sender_id references Member [member_id] reciever_id references Member [member_id]
Highest Normal Form:	4NF (at least)
Comments:	

MessageContent

message_id	INT	NOT NULL
content	VARCHAR(256)	NOT NULL
date	DATETIME	NOT NULL
read_status	VARCHAR(10)	NOT NULL

Primary Key(s):	message_id
	date
Foreign Key(s):	message_id references Message [message_id]
Highest Normal Form:	4NF
Comments:	

SportingAreaReservation

member_id	INT	NOT NULL
area_id	INT	NOT NULL
date	DATETIME	NOT NULL
check_in_time	DATETIME	
check_out_time	DATETIME	
scheduled_start	DATETIME	NOT NULL
scheduled_end	DATETIME	NOT NULL
date_canceled	DATETIME	

Primary Key(s):	member_id
	area_id
	date
Foreign Key(s):	member_id references Member [member_id]
	area_id references SportingArea [resource_id]
Highest Normal Form:	4NF (at least)
Comments:	

SportingEquipmentReservation

member_id	INT	NOT NULL
equipment_id	INT	NOT NULL
date	DATETIME	NOT NULL
check_in_time	DATETIME	
check_out_time	DATETIME	
scheduled_start	DATETIME	NOT NULL
scheduled_end	DATETIME	NOT NULL
date_canceled	DATETIME	

Primary Key(s):	member_id
	area_id
	date
Foreign Key(s):	member_id references Member [member_id]
	equipment_idreferences SportingEquipment [resource_id]
Highest Normal Form:	4NF (at least)
Comments:	

SportingCategory

category_id	INT	NOT NULL, UNIQUE
name	VARCHAR(36)	NOT NULL
description	VARCHAR(45)	
governing_staff_id	INT	NOT NULL

Primary Key(s):	category_id
Foreign Key(s):	governing_staff_id references Staff [staff_id]
Highest Normal Form:	4NF (at least)
Comments:	

Damage

equipment_id	INT	NOT NULL
date	DATETIME	NOT NULL
description	VARCHAR(45)	

Primary Key(s):	equipment_id
	date
Foreign Key(s):	equipment_id references SportingEquipment [resource_id]
Highest Normal Form:	4NF (at least)
Comments:	

Repair

area_id	INT	NOT NULL
date	DATETIME	NOT NULL
description	VARCHAR(45)	
cost	DOUBLE	

Primary Key(s):	area_id
	date
Foreign Key(s):	area_id references SportingArea [resource_id]
Highest Normal Form:	4NF
Comments:	

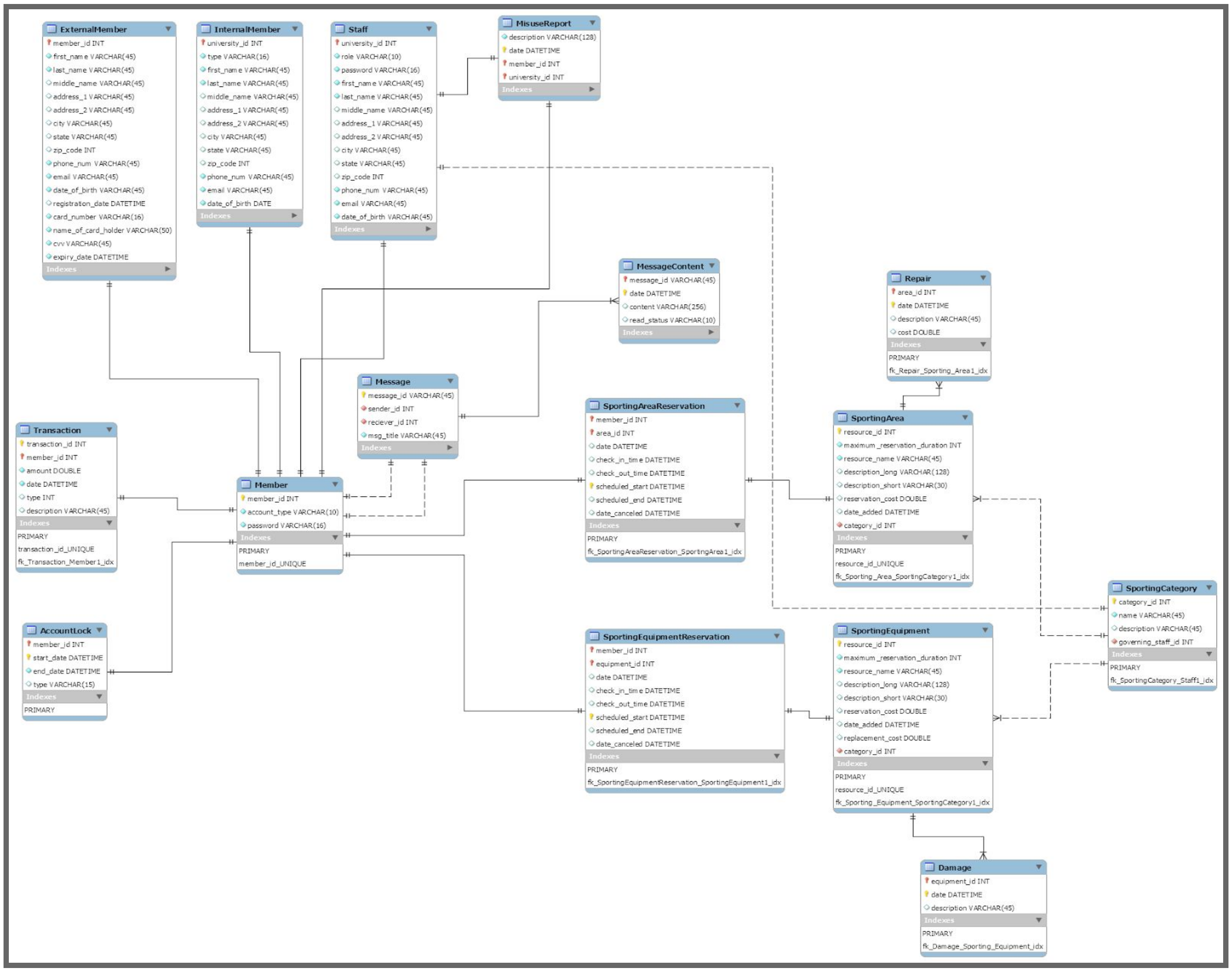
MisuseReport

member_id	INT	NOT NULL
staff_id	INT	NOT NULL
description	VARCHAR(128)	NOT NULL
date	DATETIME	NOT NULL

Primary Key(s):	member_id date
Foreign Key(s):	member_id references Member [member_id] staff_idreferences Staff [university_id]
Highest Normal Form:	4NF
Comments:	

MySQL Schema Diagram

(Actual document [possibly updated version] can be viewed with the link: here)



(Actual workbench model definition can be downloaded: here)

UNCC-SRMS Schema File (Creates Tables)

(Can be downloaded at the following link: here)

UNCC-SRMS Sample Data (Seeds the DB)

(Can be downloaded at the following link: here)

Triggers, Stored Procedures and Stored Functions

```
CREATE DEFINER=`adminUyjKkrs`@`127.10.143.130` FUNCTION
`calculate_refund_cancel_transaction`(`member_id` INT) RETURNS decimal(10,0)

  READS SQL DATA

 COMMENT 'calculates the refund amount while cancellation of transaction'

begin

        declare refund_amount decimal default 0.0;


        select t.amount*0.7 into @refund_amount

        from Transaction t,SportingAreaReservation s where

TIMESTAMPDIFF(HOUR,s.date_canceled,s.check_in_time)>24

and t.member_id=s.member_id;

return @refund_amount;

end
```

```
CREATE DEFINER=`adminUyjKkrs`@`127.10.143.130` PROCEDURE `cancel_equip_proc`(IN `mem_id`
INT, IN `e_id` INT)

BEGIN

Update SportingEquipmentReservation

set date_canceled=now()

where member_id=mem_id and equipment_id=e_id;

Update SportingEquipment

set availability=availability+1

where resource_id=a_id;

END
```

```
CREATE DEFINER=`adminUyjKkrs`@`127.10.143.130` PROCEDURE
`cancel_proc_sportingarereservartion`(IN `mem_id` INT, IN `a_id` INT)

BEGIN

Update SportingAreaReservation

set date_canceled=now()

where member_id=mem_id and area_id=a_id;

Update SportingArea

set availability='yes'

where resource_id=a_id;

END
```

```
CREATE DEFINER=`adminUyjKkrs`@`127.10.143.130` PROCEDURE `category_disp`(IN `c_id` INT)

   NO SQL

SELECT resource_name, availability

FROM SportingArea

WHERE category_id =1
```

```
UNION ALL

SELECT resource_name, availability

FROM SportingEquipment

WHERE category_id =1



CREATE DEFINER=`adminUyjKkrs`@`127.10.143.130` PROCEDURE `check_in_area_proc`(IN `mem_id`
INT, IN `a_id` INT)

BEGIN

Update SportingAreaReservation

set check_in_time= Now()

where member_id=mem_id and area=a_id;

END



CREATE DEFINER=`adminUyjKkrs`@`127.10.143.130` PROCEDURE `check_out_area_proc`(IN `mem_id`
INT, IN `a_id` INT)

BEGIN

Update SportingAreaReservation

set check_out_time=now()

where member_id=mem_id and area_id=a_id;

END

CREATE DEFINER=`adminUyjKkrs`@`127.10.143.130` PROCEDURE `check_out_equipment_proc`(in
mem_id int, in e_id int)

BEGIN

Update SportingEquipmentReservation

set check_out_time=now()

where member_id=mem_id and equipment_id=e_id;

END
```

```
CREATE DEFINER=`adminUyjKkrs`@`127.10.143.130` PROCEDURE `num_misuse`(in  mem_id int,in
a_id int, in s_end DATETIME, s_start DATETIME)

BEGIN

INSERT INTO SportingAreaReservation( area_id,date,member_id,schedule_start,schedule_end)
VALUES ( a_id,currdate(),s_start,s_end);

Update SportingArea

set availability='No'

where resource_id=a_id;

END


CREATE DEFINER=`adminUyjKkrs`@`127.10.143.130` PROCEDURE `num_msg`(in mem_id int)

BEGIN

Select count(*) from MisuseReport as mr

Where mr.member_id=mem_id;

END


CREATE DEFINER=`adminUyjKkrs`@`127.10.143.130` PROCEDURE `num_report`(IN `mem_id` INT)

BEGIN

SELECT count( * )

FROM MisuseReport

WHERE `MisuseReport`.`member_id` = mem_id;

END

CREATE DEFINER=`adminUyjKkrs`@`127.10.143.130` PROCEDURE `num_reservations`( IN memid INT)

BEGIN

SELECT (

SELECT count( * )

FROM SportingAreaReservation where SportingAreaReservation.member_id=memid

) + (

SELECT count( * )
FROM SportingEquipmentReservation where SportingEquipmentReservation.member_id=memid )  AS
num_reservation;

END

CREATE DEFINER=`adminUyjKkrs`@`127.10.143.130` PROCEDURE `reserve`(IN `mem_id` INT, IN `a_id`
INT, IN `s_end` DATETIME, IN `s_start` DATETIME)

BEGIN

INSERT INTO SportingAreaReservation( area_id,date,member_id,schedule_start,schedule_end)
VALUES ( a_id,now(),s_start,s_end);

Update SportingArea

set availability='No'

where resource_id=a_id;

END

CREATE DEFINER=`adminUyjKkrs`@`127.10.143.130` PROCEDURE `reserve_equipment`(IN `mem_id`
INT, IN `e_id` INT, IN `s_end` DATETIME, IN `s_start` DATETIME)

BEGIN

INSERT INTO SportingEquipmentReservation(
equipment_id,date,member_id,schedule_start,schedule_end) VALUES ( e_id,now(),s_start,s_end);

Update SportingEquipment

set availability=availability-1

where resource_id=e_id;

END
```

```
CREATE DEFINER=`adminUyjKkrs`@`127.10.143.130` PROCEDURE `reserve_equipment`(IN `mem_id`
INT, IN `e_id` INT, IN `s_end` DATETIME, IN `s_start` DATETIME)

BEGIN

INSERT INTO SportingEquipmentReservation(
equipment_id,date,member_id,schedule_start,schedule_end) VALUES ( e_id,now(),s_start,s_end);

Update SportingEquipment

set availability=availability-1

where resource_id=e_id;

END
```

```
CREATE DEFINER=`adminUyjKkrs`@`127.10.143.130` FUNCTION `validate_member`(`eid`
VARCHAR(45), `pwd` VARCHAR(16)) RETURNS tinyint(1)
   READS SQL DATA
   COMMENT 'Validates the member'
BEGIN
        DECLARE valid BOOLEAN;
        DECLARE mid INT;
        SELECT ID FROM `All_Members` WHERE Email_ID=eid INTO @mid;
        SELECT EXISTS (SELECT * FROM `Member` WHERE member_id=@mid AND password=pwd)
INTO @valid;
        RETURN @valid;
END
```

```
CREATE TRIGGER incr_exmemid_trig
before insert on Member
for each row
set NEW.member_id=(select max(member_id)+1 from Member where account_type='external');
CREATE TRIGGER no_del_t
BEFORE DELETE ON Member
FOR EACH ROW
SIGNAL SQLSTATE '77777'
SET MESSAGE_TEXT = 'Rows cannot be deleted from this table';

CREATE TRIGGER no_del_reserv_t
BEFORE DELETE ON SportingAreaReservation
FOR EACH ROW
SIGNAL SQLSTATE '77777'
SET MESSAGE_TEXT = 'Rows cannot be deleted from this table';

CREATE TRIGGER no_del_trnx_t
```

```
BEFORE DELETE ON Transaction

FOR EACH ROW

SIGNAL SQLSTATE '77777'

SET MESSAGE_TEXT = 'Rows cannot be deleted from this table';



CREATE TRIGGER no_del_equip_t

BEFORE DELETE ON SportingEquipmentReservation

FOR EACH ROW

SIGNAL SQLSTATE '77777'

SET MESSAGE_TEXT = 'Rows cannot be deleted from this table';
```

Indexes

```
create unique index mem_id_idx
on Member (member_id);


create unique index int_mem_idx
on InternalMember(university_id);


create unique index ext_mem_idx
on ExternalMember(member_id);

create unique index staff_idx
on Staff(university_id);

create index last_name_idx
on ExternalMember(last_name);

create index sport_ar_idx
on SportingArea(resource_id,category_id);

create index sp_equip_idx
on SportingEquipment (resource_id);

create index msg_content_idx
on MessageContent(associated_message_id);

create index msg_idx
on Message(message_id);
```

Views

```
Structure for view `All_Members`
--
DROP TABLE IF EXISTS `All_Members`;

CREATE ALGORITHM=UNDEFINED DEFINER=`adminUyjKkrs`@`127.10.143.130` SQL
SECURITY DEFINER VIEW `All_Members` AS select `InternalMember`.`university_id` AS
`ID`,concat(`InternalMember`.`first_name`,' ',`InternalMember`.`last_name`) AS
`Name`,`InternalMember`.`email` AS `Email_ID`,`InternalMember`.`phone_num` AS `Phone
No.`,`InternalMember`.`type` AS `Type` from `InternalMember` union all select
`ExternalMember`.`member_id` AS `ID`,concat(`ExternalMember`.`first_name`,'
',`ExternalMember`.`last_name`) AS `Name`,`ExternalMember`.`email` AS
`Email_ID`,`ExternalMember`.`phone_num` AS `Phone No.`,(select 'external') AS `Type` from
`ExternalMember`;

-- --------------------------------------------------------

--
-- Structure for view `equip_available`
--
DROP TABLE IF EXISTS `equip_available`;

CREATE ALGORITHM=UNDEFINED DEFINER=`adminUyjKkrs`@`127.10.143.130` SQL
SECURITY DEFINER VIEW `equip_available` AS select `SportingEquipment`.`resource_name`
AS `equipment` from `SportingEquipment` where (`SportingEquipment`.`availability` > 0);

-- --------------------------------------------------------

--
-- Structure for view `resource_avail`
--
DROP TABLE IF EXISTS `resource_avail`;

CREATE ALGORITHM=UNDEFINED DEFINER=`adminUyjKkrs`@`127.10.143.130` SQL
SECURITY DEFINER VIEW `resource_avail` AS select `SportingArea`.`resource_name` AS
`resources` from `SportingArea` where (`SportingArea`.`availability` = 'yes');

-- --------------------------------------------------------

--
-- Structure for view `Sport_Description`
```

```
--
DROP TABLE IF EXISTS `Sport_Description`;

CREATE ALGORITHM=UNDEFINED DEFINER=`adminUyjKkrs`@`127.10.143.130` SQL
SECURITY DEFINER VIEW `Sport_Description` AS select `se`.`resource_name` AS
`Equipment Name`,`sc`.`name` AS `Sport Category`,`sa`.`description_short` AS `Area
Description`,`se`.`description_short` AS `Equipment Description` from ((`SportingEquipment`
`se` join `SportingCategory` `sc`) join `SportingArea` `sa` on(((`se`.`category_id` =
`sc`.`category_id`) and (`sa`.`resource_id` = `se`.`resource_id`)))) group by
`se`.`resource_name` order by 2,1;
```