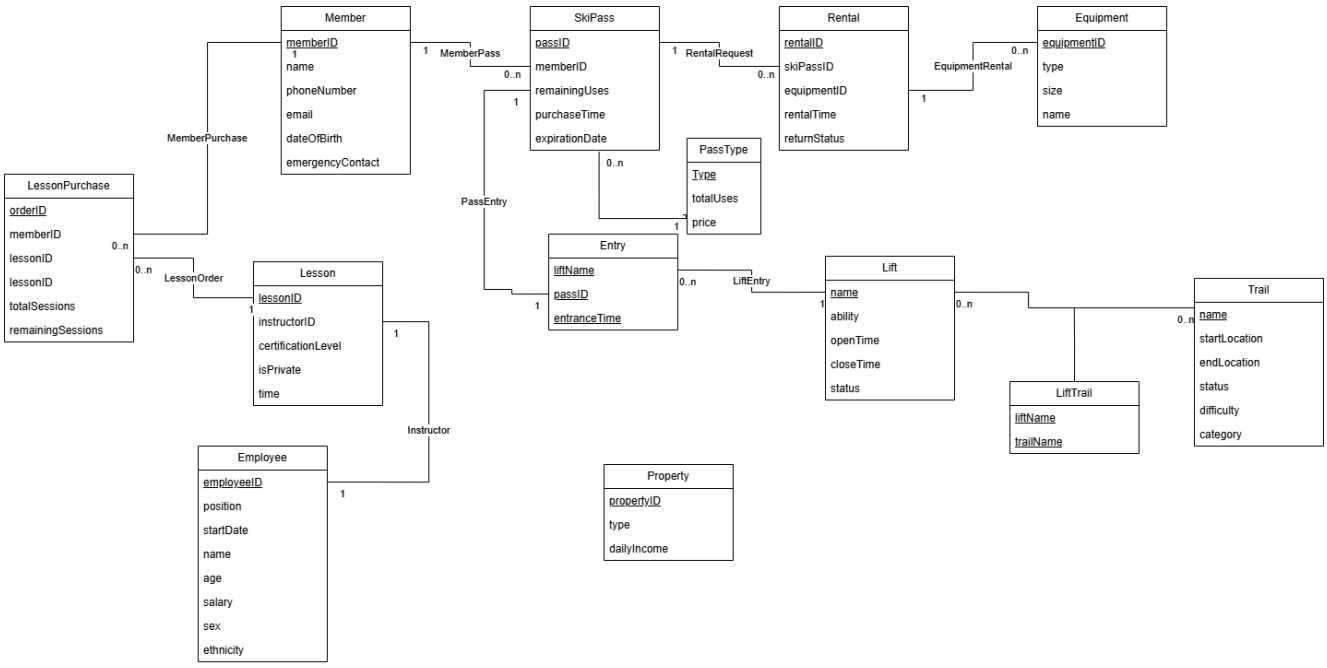
ER Diagram / Conceptual database design



This data model is our plan to manage the principal activities of a ski resort by organizing information about members, ski passes, rentals, lessons, lift usage, and property revenue. At the center is the Member entity, which contains personal data and is associated with multiple ski passes. Each SkiPass tracks the number of uses left, the date purchased, and the expiration date. Passes are tracked whenever a guest enters a lift through the Entry table, which links passes to lifts and timestamps their use. The Lift and Trail tables share a many-to-many relationship so that each lift can service multiple trails and each Trail can be serviced by multiple lifts, convenient for mapping terrain coverage and guest traffic. Rentals are handled separately with a Rental table correlating ski passes to specific Equipment to facilitate tracking of inventory and equipment return. Lessons are handled with a flexible model, members can purchase multi session lessons via the LessonPurchase table, with the Lesson table containing details such as level, instructor, and time. Instructors are pulled from the Employee table, which also serves more general HR needs like salary, position, and demographics. Finally, the Property table allows the resort to track income generated by amenities like lodges or cabins.

Logical database design and Normalization Analysis

Member

Member_ID	name	phone	email	date_of_birth	emergency_c contact

PK: MemberID
CK: email
FDs:
1. member_id → name, phone, email, date_of_birth, emergency_contact
2. email → member_id, name, phone, date_of_birth, emergency_contact (since email is unique)
3NF: member_id and email are superkeys and there are no transitive dependencies.

PassType

type	Total_uses	price

PK:type
FDs:
1. type → total_uses, price

3NF: Yes. The only FD has the PK (superkey) on the left.

SkiPass

Pass_ID	Member_ ID	remaining_ uses	purchase_time	expiration_date

PK: Pass_ID
FK: Member_ID
CK:
FDs:
1. pass_id → member_id, type, remaining_uses, purchase_time, expiration_date

3NF: Yes. The only FD has the PK (superkey) on the left.

SkiPass_Archive:

SParchive_id	pass_id	member_id	type	remaining_uses	purchase_time	expiration_date	archived_time

PK: SParchive_id
FK: pass_id, member_id, type
CK:
FDs:

1. SParchive_id → pass_id, member_id, type, remaining_uses, purchase_time, expiration_date, archived_time

3NF: Yes. The only FD's X is the PK.

Trail:

trail_name	start_location	end_location	status	difficulty	category

PK: Trail_name
FDs:

1. trail_name → start_location, end_location, status, difficulty, category

3NF: Yes. The left side of the FD is a superkey.

Lift:

lift_name	ability	open_time	close_time	status

PK:lift_name
FDs:

1. lift_name → ability, open_time, close_time, status

3NF: Yes. The left side of the FD is a superkey in the FD.

LiftTrial:

liftName	TrailName

PK:(lift_name, trail_name)
FK: liftname, trailname
FDs:

1. (lift_name, trail_name) → (no other columns)

3NF/BCNF: Yes (no non-key columns).

Entry:

lift_name	pass_id	entrance_time

PK: (lift_name, pass_id, entrance_time)
FK: liftname, pass_id
FDs:

1. (lift_name, pass_id, entrance_time) → (no other columns)

3NF: Yes (no non-key columns).

Property:

propertyID	property_type	daily_profit

PK: propertyID
FDs:

1. propertyID → {proptery_type, daily_income}

3NF: Yes since the only X side is the primary key

Rental:

rentalID	skiPassID	equipmentID	rentalTime	returnStatus

PK: rentalID
FK: {skiPassID,equipmentID}
FDs:

1) rentalID → {skiPassID,equipmentID,rentalTime,returnStatus}

3NF: Yes, only FD's LHS (X in X→Y) is a superkey as it contains the primary key, which is also a candidate key.

NOTE: For this relation, its archival relation is Rental_Archive, which contains all the above fields plus an archive ID, an archive time, and a changeState field. The changeState can either be 0 (indicating a record creation event), 1 (indicating a record update), or 2 (indicating a record deletion). Furthermore, since an admin can override the rental time on a rental record, the only FD in this archive is archived → {ALL}, so this archive is also technically in 3NF.

Equipment:

equipmentID	equip_type	equip_size	name

PK: equipmentID
FDs:

1) equipmentID→{equip_type,equip_size,name}

3NF: Yes, only FD's X is a superkey as it is a primary key, which automatically means it's a candidate key, and thus X fulfills the definition of a superkey as it contains a CK.

NOTE ONE: There are a couple of key constraints on this relation…

1) For the equip_type field, valid equipment types are restricted to those contained in the following set: {'boot', 'pole', 'snowboard', 'alpine ski', 'helmet', 'goggle', 'glove'}. Thus, in the user application, when one enters an equipment type not contained within this set, they will receive a constraint error. Also, I included the three ladder items to address the protective gear rental equipment.
2) For each equipment type, a valid equipment size is limited within a certain range. From the spec: "Ski boots range from size 4 to 14 with half sizes, the length of ski poles ranges from 100cm to 140cm (integer), the length of ski ranges from 115cm to 200cm (integer), and the length of snowboard ranges from 90cm to 178cm (integer)". Additionally, valid sizes for protective equipment are 1 (small), 2(medium), 3(large).

NOTE TWO: Just like the rental archive above, the Equipment_Archive contains all the fields of the equipment relation, plus an archive ID, an archive time, and a changeState. The change state field is identical to the same field of the rental archive.

HUGELY IMPORTANT NOTE FOR RENTAL AND EQUIPMENT: The admin password to change equipment fields and the time of a rental is "1234" as set at the very top of the SkiResort class.

Employee

employee_id	position	start_date	name	age	salary	sex	ethnicity	certification_level

PK: employee_id
FDs:

1) {employee_id} -> {position, start_date, name, age, sex, ethnicity, certification_level}
2) {employee_id, position, start_date, certification_level} -> {salary}

3NF: Yes, the FDs X is a super key, as it contains the primary key. We determined that the only potential issue is salary, but that can be determined per employee through bonuses, raises, and promotions that may not be exclusively tied to the other values.

Lesson

lesson_id	instructor_id	private	time

PK: lesson_id
FDs:
1) {lesson_id} -> {instructorID, private, time}

3NF: Yes, only FD's X is a super key as it is a primary key.

LessonPurchase

order_id	member_id	lesson_id	total_sessions	remaining_sessions

PK: order_id
FDs:
1) {order_id} -> {member_id, lesson_id, total_sessions, remaining_sessions}

3NF: Yes, only FD's X is a super key as it is a primary key.
NOTE: This relation has LessonPurchase_Archive, which has the same schema except it is missing the remaining_sessions column.

Query Description:

Our custom query answers the question of what the total profit of the Ski Resort will be for a requested number of years with the current daily incomes from the properties minus the current salaries of all employees. The user is also prompted to enter how many days of the year the ski season is estimated to last since the properties will likely only be making their normal profit during this time. This number is used to multiply the daily profits by that many days before subtracting the salaries. This is a useful tool for the business side of the ski resort so they can track the current pace for the money they will make, as all business owners want to do.