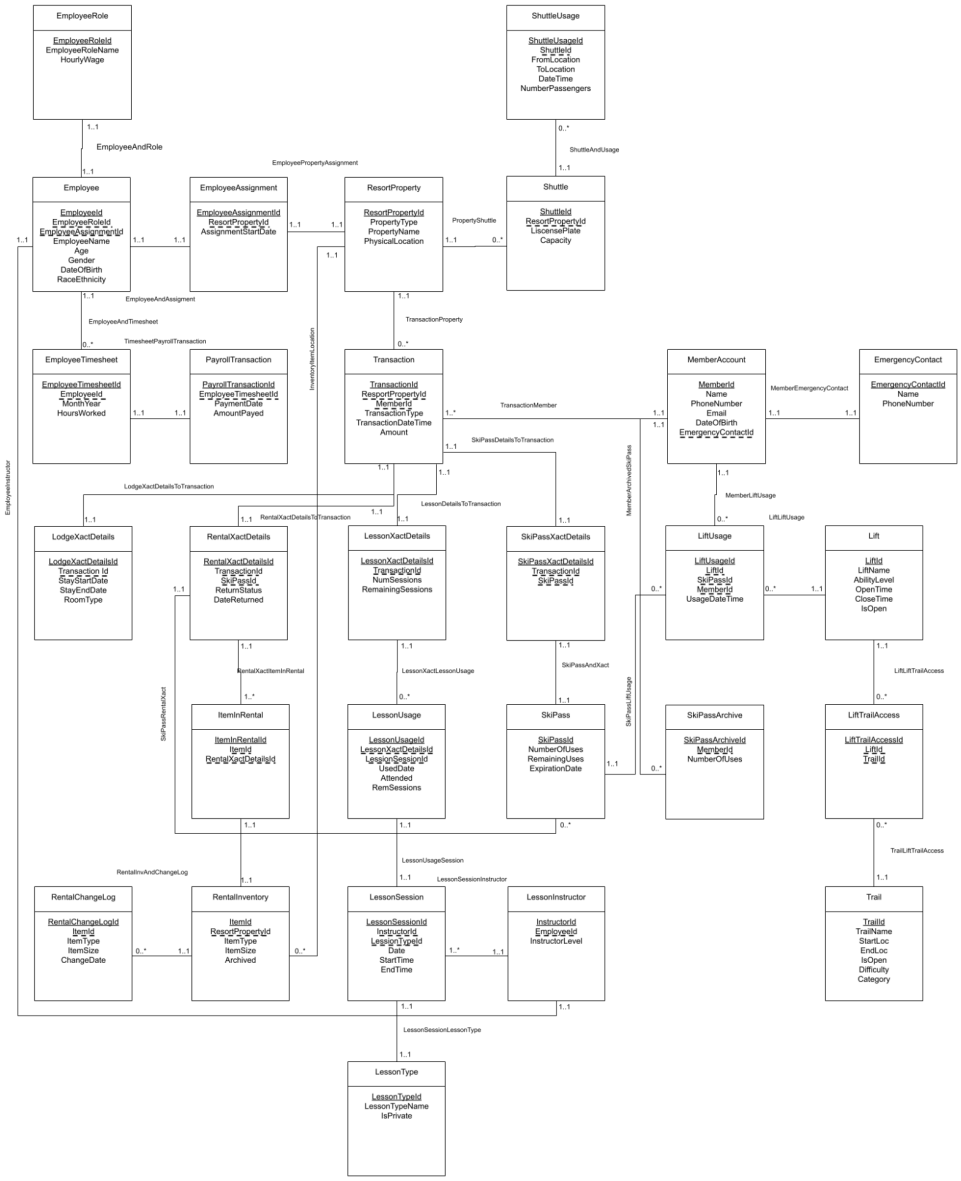
CSC 460 Final Project

5/4/2025

Seth Jernigan and Marco Pena

Conceptual Database Design

Below is the E-R diagram (in UML form) for our Ski Resort database.



This E-R diagram is quite complex, and there are a variety of relationships that require more extensive description to understand our database's conceptual structure. To begin with, the ResortProperty table holds information about each building or establishment within the ski resort, including lodges, gift shops, rental centers, visitor centers, ski schools, parking lots, shuttle hubs, etc. Employees are assigned to work at exactly one of these locations. Transactions are also linked to one of these locations to enable understanding of transactions within each location.

Continuing on with the concept of transactions, there exists an overall Transaction table which contains information that every monetary transaction carries with it, and its transaction types. For some transaction types, such as lodge stays, rentals, lesson purchases, and ski pass purchases, more details are associated with the transaction, and separate tables are created for these details. Rentals also have some intricacies. Each rental transaction can be associated with 1 or more rental items from the RentalInventory table through the ItemInRental table. Additionally, changes in rental items are logged in RentalChangeLog. Lesson transactions are associated with specific lesson sessions through the LessonUsage table. Lesson sessions have an associated instructor as well as a specific lesson type from the LessonType table. Ski Pass transactions must be associated with an already existing ski pass, and a SkiPassArchive table exists for holding deleted ski passes.

Finally, a few intricacies exist surrounding the Employee relation. Specifically, employees are associated with an 'assignment', that is, where they work in the resort. They are also associated with a 'role', which specifies what they do and their pay range. This employee role associates with the EmployeeTimesheet relation, which allows proper payment amounts within the PayrollTransaction relation.

Logical Database Design

The following relational schemas define the logical database design for our Ski Resort database:

EMPLOYEE-RELATED TABLES

Table: EmployeeRole

- employeeRoleId (Primary Key) (Integer)
- employeeRoleName (String)
- hourlyWage (Decimal)

Table: Employee

- employeeId (Primary Key) (Integer)
- employeeRoleId (Foreign Key → EmployeeRole.employeeRoleId) (Integer)
- employeeAssignmentId (Foreign Key → EmployeeAssignment.employeeAssignmentId) (Integer)
- employeeName (String)
- age (Integer)
- gender (String)
- dateOfBirth (Date)
- raceEthnicity (String)

Table: EmployeeAssignment

- employeeAssignmentId (Primary Key) (Integer)
- resortPropertyId (Foreign Key → ResortProperty.resortPropertyId) (Integer)

- assignmentStartDate (Date)

Table: EmployeeTimesheet

- employeeTimesheetId (Primary Key) (Integer)
- employeeId (Foreign Key → Employee.employeeId) (Integer)
- monthYear (Date)
- hoursWorked (Decimal)

Table: PayrollTransaction

- payrollTransactionId (Primary Key) (Integer)
- employeeTimesheetId (Foreign Key → EmployeeTimesheet.employeeTimesheetId) (Integer)
- paymentDate (Date)
- amountPayed (Decimal)

RESORT PROPERTY TABLES

Table: ResortProperty

- resortPropertyId (Primary Key) (Integer)
- propertyType (String – must be one of: LODGE, GIFT_SHOP, RENTAL_CENTER, VISITOR_CENTER, SKI_SCHOOL, FREE_PARKING, PAID_PARKING)
- propertyName (String)
- physicalLocation (String)

Table: Shuttle

- shuttleId (Primary Key) (Integer)
- resortPropertyId (Foreign Key → ResortProperty.resortPropertyId) (Integer)
- licensePlate (String)
- capacity (Integer)

Table: ShuttleUsage

- shuttleUsageId (Primary Key) (Integer)
- shuttleId (Foreign Key → Shuttle.shuttleId) (Integer)
- fromLocation (String)
- toLocation (String)
- dateTime (Timestamp)
- numberPassengers (Integer)

MEMBER TABLES

Table: EmergencyContact

- emergencyContactId (Primary Key) (Integer)
- name (String)
- phoneNumber (String)

Table: MemberAccount

- memberId (Primary Key) (Integer)
- name (String)
- phoneNumber (String)

- email (String)
- dateOfBirth (Date)
- emergencyContactId (Foreign Key → EmergencyContact.emergencyContactId) (Integer)

SKI PASS TABLES

Table: SkiPass

- skiPassId (Primary Key) (Integer)
- numberOfUses (Integer)
- remainingUses (Integer)
- expirationDate (Date)

Table: SkiPassArchive

- skiPassArchiveId (Primary Key) (Integer)
- memberId (Foreign Key → MemberAccount.memberId) (Integer)
- numberOfUses (Integer)

Table: Lift

- liftId (Primary Key) (Integer)
- liftName (String)
- abilityLevel (String – beginner/intermediate/expert)
- openTime (String)
- closeTime (String)
- isOpen (Boolean – 0 or 1)

Table: Trail

- trailId (Primary Key) (Integer)
- trailName (String)
- startLoc (String)
- endLoc (String)
- isOpen (Boolean – 0 or 1)
- difficulty (String – beginner/intermediate/expert)
- category (String – groomed, park, moguls, glade)

Table: LiftUsage

- liftUsageId (Primary Key) (Integer)
- liftId (Foreign Key → Lift.liftId) (Integer)
- skiPassId (Foreign Key → SkiPass.skiPassId) (Integer)
- memberId (Foreign Key → MemberAccount.memberId) (Integer)
- usageDateTime (Timestamp)

Table: LiftTrailAccess

- liftTrailAccessId (Primary Key) (Integer)
- liftId (Foreign Key → Lift.liftId) (Integer)
- trailId (Foreign Key → Trail.trailId) (Integer)

RENTAL TABLES

Table: RentalInventory

- itemId (Primary Key) (Integer)
- resortPropertyId (Foreign Key → ResortProperty.resortPropertyId) (Integer)
- itemType (String – ski, snowboard, etc.)
- itemSize (String)
- archived (Boolean – 0 or 1)

Table: RentalChangeLog

- rentalChangeLogId (Primary Key) (Integer)
- itemId (Foreign Key → RentalInventory.itemId) (Integer)
- itemType (String)
- itemSize (String)
- changeDate (Date)

Table: ItemInRental

- itemInRentalId (Primary Key) (Integer)
- itemId (Foreign Key → RentalInventory.itemId) (Integer)
- rentalXactDetailsId (Foreign Key → RentalXactDetails.rentalXactDetailsId) (Integer)

LESSON & INSTRUCTOR TABLES

Table: LessonInstructor

- instructorId (Primary Key) (Integer)
- employeeId (Foreign Key → Employee.employeeId) (Integer)

- instructorLevel (String – Level I, II, III)

Table: LessonType

- lessonTypeId (Primary Key) (Integer)
- lessonTypeName (String)

  isPrivate (Boolean – 0 for group, 1 for private)

Table: LessonSession

- lessonSessionId (Primary Key) (Integer)
- instructorId (Foreign Key → LessonInstructor.instructorId) (Integer)
- lessonTypeId (Foreign Key → LessonType.lessonTypeId) (Integer)
- sessionDate (Date)
- startTime (String)
- endTime (String)

Table: LessonUsage

- lessonUsageId (Primary Key) (Integer)
- lessonXactDetailsId (Foreign Key → LessonXactDetails.lessonXactDetailsId) (Integer)
- lessonSessionId (Foreign Key → LessonSession.lessonSessionId) (Integer)
- usedDate (Date)
- attended (Boolean – 0 or 1)
- remSessions (Integer)

TRANSACTIONS TABLES

Table: Transactions

- transactionId (Primary Key) (Integer)
- resortPropertyId (Foreign Key → ResortProperty.resortPropertyId) (Integer)
- memberId (Foreign Key → MemberAccount.memberId) (Integer)
- transactionType (String)
- transactionDateTime (Timestamp)
- amount (Decimal)

Table: RentalXactDetails

- rentalXactDetailsId (Primary Key) (Integer)
- transactionId (Foreign Key → Transactions.transactionId) (Integer)
- skiPassId (Foreign Key → SkiPass.skiPassId) (Integer)
- returnStatus (Boolean – 0 if not returned, 1 if returned)
- dateReturned (Date)

Table: LodgeXactDetails

- lodgeXactDetailsId (Primary Key) (Integer)
- transactionId (Foreign Key → Transactions.transactionId) (Integer)
- stayStartDate (Date)
- stayEndDate (Date)
- roomType (String)

Table: LessonXactDetails

- lessonXactDetailsId (Primary Key) (Integer)
- transactionId (Foreign Key → Transactions.transactionId) (Integer)
- numSessions (Integer)
- remainingSessions (Integer)

Table: SkiPassXactDetails

- skiPassXactDetailsId (Primary Key) (Integer)
- transactionId (Foreign Key → Transactions.transactionId) (Integer)
- skiPassId (Foreign Key → SkiPass.skiPassId) (Integer)

Normalization Analysis

Each of our relations adhere to 3NF. Below, this is shown to be the case for all tables.

Table: Employee Role

Functional Dependencies: employeeRoleId ⇒ employeeRoleName, hourlyWage

Proof: This is in 3NF since employeeRoleId is the primary key, and is thus is a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: Employee

Functional Dependencies: employeeId ⇒ employeeRoleId, employeeAssignmentId, employeeName, age, gender, dateOfBirth, raceEthnicity

Proof: By definition, this is in 3NF since employeeId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: EmployeeTimesheetId

Functional Dependencies: employeeTimesheetId ⇒ employeeId, monthYear, hoursWorked

Proof: By definition, this is in 3NF since employeeTimesheetId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: PayrollTransaction

Functional Dependencies: payrollTransactionId ⇒ employeeTimesheetId, paymentDate, amountPayed

Proof: By definition, this is in 3NF since payrollTransactionId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: ResortProperty

Functional Dependencies: resortPropertyId ⇒ propertyType, propertyName, physicalLocation

Proof: By definition, this is in 3NF since resortPropertyId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: Shuttle

Functional Dependencies: shuttleId ⇒ resortPropertyId, licensePlate, capacity, licensePlate ⇒ capacity

Proof: By definition, this is in 3NF since shuttleId is the primary key, and is thus a superkey of the relation, and also since licensePlate is a candidate key, therefore is a superkey. By the same arguments, by definition, this table is in BCNF.

Table: ShuttleUsage

Functional Dependencies: shuttleUsageId ⇒ shuttleId, fromLocation, toLocation, dateTime, numberPassengers

Proof: By definition, this is in 3NF since shuttleUsageId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: EmergencyContact

Functional Dependencies: emergencyContactId ⇒ name, phoneNumber

Proof: By definition, this is in 3NF since emergencyContactId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: MemberAccount

Functional Dependencies: memberId ⇒ name, phoneNumber, email, dateOfBirth, emergencyContactId

Proof: By definition, this is in 3NF since memberId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: SkiPass

Functional Dependencies: skiPassId ⇒ numberOfUses, remainingUses, expirationDate

Proof: By definition, this is in 3NF since skiPassId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: SkiPassArchive

Functional Dependencies: skiPassArchiveId ⇒ memberId, numberOfUses

Proof: By definition, this is in 3NF since skiPassArchiveId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: Lift

Functional Dependencies: liftId ⇒ liftName, abilityLevel, openTime, closeTime, isOpen

Proof: By definition, this is in 3NF since liftId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: Trail

Functional Dependencies: trailId ⇒ trailName, startLoc, endLoc, isOpen, difficulty, category

Proof: By definition, this is in 3NF since trailId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: RentalInventory

Functional Dependencies: itemId ⇒ resortPropertyId, itemType, itemSize, archived

Proof: By definition, this is in 3NF since itemId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: RentalChangeLog

Functional Dependencies: rentalChangeLogId ⇒ itemId, itemType, itemSize, changeDate

Proof: By definition, this is in 3NF since rentalChangeLogId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: LessonInstructor

Functional Dependencies: instructorId ⇒ employeeId, instructorLevel, employeeId ⇒ instructorLevel, employeeId ⇒ instructorId

Proof: By definition, this is in 3NF since instructorId is the primary key, and is thus a superkey of the relation, and similarly, employeeId is a candidate key, meaning it is a superkey. By the same arguments, by definition, this table is in BCNF.

Table: LessonType

Functional Dependencies: lessonTypeId ⇒ lessonTypeName, isPrivate

Proof: By definition, this is in 3NF since lessonTypeId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: LessonSession

Functional Dependencies: lessonSessionId ⇒ instructorId, lessonTypeId, sessionDate, startTime, endTime

Proof: By definition, this is in 3NF since lessonSessionId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: EmployeeAssignment

Functional Dependencies: employeeAssignmentId ⇒ resortPropertyId, assignmentStartDate

Proof: By definition, this is in 3NF since employeeAssignmentId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: Transactions

Functional Dependencies: transactionId ⇒ resortPropertyId, memberId, transactionType, transactionDateTime, amount

Proof: By definition, this is in 3NF since transactionId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: RentalXactDetails

Functional Dependencies: rentalXactDetailsId ⇒ transactionId, skiPassId, returnStatus, dateReturned, transactionId => rentalXactDetailsId, skiPassId, returnStatus, dateReturned

Proof: By definition, this is in 3NF since rentalXactDetailsId is the primary key, and is thus a superkey of the relation, and similarly, transactionId is a candidate key, so it is also a superkey. By the same arguments, by definition, this table is in BCNF.

Table: LodgeXactDetails

Functional Dependencies: lodgeXactDetailsId ⇒ transactionId, stayStartDate, stayEndDate, roomType, transactionId => lodgeXactDetailsId, stayStartDate, stayEndDate, roomType

Proof: By definition, this is in 3NF since lodgeXactDetailsId is the primary key, and is thus a superkey of the relation, and similarly, transactionId is a candidate key, so it is also a superkey. By the same arguments, by definition, this table is in BCNF.

Table: LessonXactDetails

Functional Dependencies: lessonXactDetailsId ⇒ transactionId, numSessions, remainingSessions, transactionId => lessonXactDetailsId, numSessions, remainingSessions

Proof: By definition, this is in 3NF since lessonXactDetailsId is the primary key, and is thus a superkey of the relation, and similarly, transactionId is a candidate key, so it is also a superkey. By the same arguments, by definition, this table is in BCNF.

Table: SkiPassXactDetails

Functional Dependencies: skiPassXactDetailsId ⇒ transactionId, skiPassId, transactionId ⇒ skiPassXactDetailsId, skiPassId

Proof: By definition, this is in 3NF since skiPassXactDetailsId is the primary key, and is thus a superkey of the relation, and similarly, transactionId is a candidate key, so it is also a superkey. By the same arguments, by definition, this table is in BCNF.

Table: LiftUsage

Functional Dependencies: liftUsageId ⇒ liftId, skiPassId, memberId, usageDateTime

Proof: By definition, this is in 3NF since liftUsageId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: LiftTrailAccess

Functional Dependencies: liftTrailAccessId ⇒ liftId, trailId

Proof: By definition, this is in 3NF since liftTrailAccessId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: ItemInRental

Functional Dependencies: itemInRentalId ⇒ itemId, rentalXactDetailsId

Proof: By definition, this is in 3NF since itemInRentalId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Table: LessonUsage

Functional Dependencies: lessonUsageId ⇒ lessonXactDetailsId, lessonSessionId, usedDate, attended, remSessions

Proof: By definition, this is in 3NF since lessonUsageId is the primary key, and is thus a superkey of the relation. By the same argument, by definition, this table is in BCNF.

Query Description

Our fourth query that was self-designed answers the following question: For a given member, list the type of equipment used (ski or snowboard), size they typically wear, and prices they've paid. This query is useful to have handy from the perspective of the Ski Resort because this query provides a personalized analysis of a member's rental history by gathering input for a specific member ID. It joins data from multiple relations (Transactions, RentalXactDetails, ItemInRental, and RentalInventory) to determine the member's preferred type of equipment, the size they typically wear, and the prices they've paid. This adds value to the member because it is personalized for them.