Trevin Hartzler
Prof. Joseph Gradecki
Re-Design: Database
November 26, 2023

HomeEasy - Database Design

Database Technology Choice:

HomeEasy will be setup so the same front-end web application serves different clients with their own data. However, all the data will be stored in the same database. The HomeEasy web application will require a fairly complicated database to store data for many different types of objects and their associated connections. Most of the data I will be storing will be well structured data making up People, Applications, Properties, Leases, Companies, etc. and are primarily made up of text and numbers. There will be some files to be stored, but I anticipate using a 3rd party file server to manage the storage of those and then storing the fileID in the database. Additionally, there is a large need for strong relationships between data that can often change. For example, if the tenant moves from one property to another one, the user will stay the same, but the relationship through the lease will need to change. Due to these factors, I chose to use a relational database (such as SQL).

The majority of my data will be split into 2 categories. The first will be application-specific data that is not user dependent and can't be changed by the user. For example, payment methods, payment statuses, and account types will all be in app database tables. The other database tables will store user specific data. For example, login credentials, properties, leases, fee schedules, etc. would be stored in the user database tables. The user accounts to access the database server are shown below.

username	app Tables (R/W/N*)	user Tables (R/W/N*)
authAPI	N	R (userAccount), W (userSession), N (other)
tenantAPI	N	R
landlordAPI	R	W
adminAPI	W	W

*access designation = Read/Write/None

Table Uses, Attributes, and Relationships:

To designate the categories or data tables, the prefix app- and user- will be used on each table. The app-… tables will be mostly used for populating combo-box selection options. The user-… tables typically will have some tie back to the company they're aligned to.

userPeople –

Use:

The userPeople table is meant to store the name of a person, their phone number, and their address if they have one. This stores people that a renter or landlord to enters to the system. This can include applicants, co-applicants, applicant's supervisor, an applicant's reference, or a subscribed company's representative. The people data are stored independent of any company or user. When a userAccount is created, it is tied back to the personID. This allows an applicant to complete an application as a guest and then create an account later (if the given person is chosen as a tenant they will need an account).

Attributes:

Name	Type	Primary Key	Allow Nulls
personID	int	☑	☐
firstName	varchar(50)	☐	☐
lastName	varchar(50)	☐	☑
phoneNumber	char(10)	☐	☐
addressID	int	☐	☑
createdOn	datetime	☐	☐

Relationships:

The following relationships occur with the userPeople table:

Foreign Table	Foreign Column	Column	Type	Description
userPersonDetails	personID	personID	One-to-Many	One person may have many pieces of detail linked to their personID
userPersonDetails	setPersonID	personID	One-to-Many	One person can set many different pieces of detail.
userReferences	personID	personID	One-to-Many	One person can be set as many references
userEmploymentHistory	applicantID	personID	One-to-Many	One person can be linked to multiple employment history records
userEmploymentHistory	supervisorID	personID	One-to-Many	One person can be the supervisor of many different employment history records
userAddresses	addressID	addressID	Many-to-One	Many people can have the same address
userAccounts	userID	personID	One-to-One	One person can have one account
userLeasePeople	personID	personID	One-to-Many	One person can be connected to many leases. LeasePerson is interface table for a many-to-many connection between userLease and userPeople tables
userApplications	applicantID	personID	One-to-Many	One person can submit many applications
userApplications	coApplicantID	personID	One-to-Many	One person can co-apply many times

Foreign Table	Foreign Column	Column	Type	Description
userApplications	currentOwnerID	personID	One-to-Many	An applicant's current landlord could be the landlord on many applicants' applications
userApplications	previousOwnerID	personID	One-to-Many	An applicant's previous landlord could be the previous landlord on many applicants' applications

userPersonDetails –

Use:

The userPeopleDetails table is meant to store additional information about a person. This essentially allows an unlimited number of attributes to be stored about a person without changing the database table. This will allow a landlord or tenant to add items like SSN, DOB, or Middle Initial/Name. The table also stores information to track changes to the person details through the rev field as well as tracking when and who modified it.

Attributes:

Name	Type	Primary Key	Allow Nulls
personID	int	☑	☐
detailID	int	☑	☐
rev	int	☑	☐
propertyValue	varchar(50)	☐	☑
setDate	datetime	☐	☐
setPersonID	int	☐	☐

Relationships:

The following relationships exist for the userPersonDetails table:

Foreign Table	Foreign Column	Column	Type	Description
userPeople	personID	personID	Many-to-One	Many pieces of detail can be linked to one person
userDetailOptions	detailID	detailID	Many-to-One	Many people can have the same one piece of detail
userPeople	personID	setPersonID	Many-to-One	Many pieces of detail can be set by the same one person

userDetailOptions –

Use:

The userDetailOptions table stores options for additional person details properties. This allows a landlord or tenant to select additional items like SSN, DOB, or Middle Initial/Name to define in the userPersonDetails table.

Attributes:

Name	Type	Primary Key	Allow Nulls
detailID	int	☑	☐
propertyName	text	☐	☑

Relationships:

The following relationships exist for the userDetailOptions table:

Foreign Table	Foreign Column	Column	Type	Description
userPersonDetails	detailID	detailID	One-to-Many	One detail item can be set for many personDetails

userCompanyRoles –

Use:

The userCompanyRoles table is an interface table between the userCompanies table and the userPeople table to describe the role people have in the company. The roles are stored in the userAccountTypes table and are used to restrict property management employees from doing things they're not supposed to do. Also allows for many people to manage properties for many companies with switching possible between users and their companies. If a landlord hires someone, they can create an account and that new employee can interact with the properties as specified by the admin role. The admin can assign other admin, etc.

Attributes:

Name	Type	Primary Key	Allow Nulls
roleID	int	☑	☐
companyID	int	☐	☐
userID	int	☐	☐
roleTypeID	int	☐	☐
role_begin	int	☐	☐
assignedUser	int	☐	☐
role_end	int	☐	☑
endedUser	int	☐	☑

Relationships:

The following relationships exist for the userCompanyRoles table:

Foreign Table	Foreign Column	Column	Type	Description
userCompanies	companyID	companyID	Many-to-One	Many company-roles can be assigned to one company
userAccounts	userID	userID	Many-to-One	Many company-roles can be assigned to one person
appAccountTypes	accountTypeID	roleTypeID	Many-to-One	Many company-roles can have the same type
userAccounts	userID	assignedUser	Many-to-One	Many company-roles can be assigned by one person
userAccounts	userID	endedUser	Many-to-One	Many company-roles can be removed by one person

userAddresses –

Use:

The userAddresses table stores physical addresses. This allows a landlord to add an address for their property and their company's physical location as well as billing location. Tenants can add addresses for their current residence at the time of application, previous residence location, and addresses for references.

Attributes:

Name	Type	Primary Key	Allow Nulls
addressID	int	☑	☐
houseNumber	int	☐	☐
streetName	varchar(30)	☐	☐
apptNo	varchar(6)	☐	☑
city	varchar(30)	☐	☐
state	varchar(2)	☐	☐
zipCode	numeric(5,0)	☐	☐

Relationships:

The following relationships exist for the userAddresses table:

Foreign Table	Foreign Column	Column	Type	Description
userCompanies	mailingAddress	addressID	One-to-Many	One address can be linked to many companies
userCompanies	billingAddress	addressID	One-to-Many	One address can be linked to many companies
userPeople	addressID	addressID	One-to-Many	One address can be associated with many different people
userProperties	addressID	addressID	One-to-Many	One address could be associated with many properties (if for example the property was transferred to a different owner and a new property record was created)
userApplications	currentAddressID	addressID	One-to-Many	One address can be a current address on many applications
userApplications	previousAddressID	addressID	One-to-Many	One address can be a previous address on many applications

userCompanies –

Use:

The userCompanies table stores information related to a landlord's company. Every landlord must have a company recorded (even if it's informal, it counts). The company is responsible for the subscription payment and data is stored/populated based on each company. Tenants supply the company with information for applications and companies supply tenants with information about rent payments and lease agreements.

Attributes:

Name	Type	Primary Key	Allow Nulls
companyID	int	☑	☐
companyName	varchar(50)	☐	☐
phoneNumber	char(10)	☐	☐
mailingAddress	int	☐	☐
billingAddress	int	☐	☑
emailInvoiceAddress	varchar(50)	☐	☑
EIN	int	☐	☑
createdBy	int	☐	☐
createDate	datetime	☐	☐

Relationships:

The following relationships exist for the userCompanies table:

Foreign Table	Foreign Column	Column	Type	Description
userAddresses	addressID	mailingAddress	Many-to-One	Many companies can have the same mailing address
userAddresses	addressID	billingAddress	Many-to-One	Many companies can have the same billing address
userProperties	companyID	companyID	One-to-Many	One company can have many properties
userCompanyRoles	companyID	companyID	One-to-Many	One company can have many company-roles
userAccounts	userID	createdBy	Many-to-One	Many companies can be created by the same user

userProperties –

Use:

The userProperties table stores information about a company's properties. A landlord must enter the details about the properties to make it available for tenants to apply to.

Attributes:

Name	Type	Primary Key	Allow Nulls
propertyID	int	☑	☐
companyID	int	☐	☐
addressID	int	☐	☐
bedroomCount	int	☐	☐
bathroomCount	float	☐	☐
parkingCount	int	☐	☑
garageCount	int	☐	☑
storiesCount	float	☐	☑
homeType	varchar(50)	☐	☐
yearBuilt	numeric(4,0)	☐	☑
purchasePrice	int	☐	☑
purchaseDate	date	☐	☑
schoolDistrict	varchar(50)	☐	☐
nickname	varchar(50)	☐	☐
createUser	int	☐	☐
createDate	datetime	☐	☐

Relationships:

The following relationships exist for the userProperties table:

Foreign Table	Foreign Column	Column	Type	Description
userAddresses	addressID	addressID	Many-to-One	For now, many properties could be associated with one address (if for example the property was bought by a different owner)

Foreign Table	Foreign Column	Column	Type	Description
userCompanies	companyID	companyID	Many-to-One	Many properties can be linked to the same company
userLease	propertyID	propertyID	One-to-Many	One property can have many leases
userAccounts	userID	createUser	Many-to-One	Many properties can be created by the same users

userLeases –

Use:

The userLeases table stores information about leases a landlord sets up. A landlord must enter the details about the lease terms and connect it to an available property. The tenant will be able to view details regarding leases they are currently connected to as well as apply to new leases.

Attributes:

Name	Type	Primary Key	Allow Nulls
leaseID	int	☑	☐
propertyID	int	☐	☐
leaseStatus	varchar(25)	☐	☑
availableDate	date	☐	☐
moveInDate	date	☐	☑
terminationDate	date	☐	☑
leaseOccurrence	int	☐	☐
leaseSuccessionOccurr...	int	☐	☑
securityDeposit	float	☐	☐
contractDocID	varchar(50)	☐	☑
createUser	int	☐	☐
createDate	datetime	☐	☐

Relationships:

The following relationships exist for the userLeases table:

Foreign Table	Foreign Column	Column	Type	Description
userProperties	propertyID	propertyID	Many-to-One	Many leases can be connected to one property
userDependants	leaseID	leaseID	One-to-Many	One lease can have many minors who are dependent on the leaseholder(s)
userLeasePeople	leaseID	leaseID	One-to-Many	One lease can have many people on the lease.
userLeaseFees	leaseID	leaseID	One-to-Many	One lease can have many lease fees associated with it

Foreign Table	Foreign Column	Column	Type	Description
userLeaseFees	leaseID	leaseID	One-to-Many	One lease can have many lease fees associated with it
userAccounts	userID	createUser	Many-to-One	Many leases can be created by the same user
userApplications	applicationID	applicationID	One-to-Many	One lease can have many lease applications
appOccurrences	occurrenceID	leaseOccurrence	Many-to-One	Many leases can have the same occurrence
appOccurrences	occurrenceID	leaseSuccession Occurrence	Many-to-One	Many leases can have the same occurrence behavior after the primary lease period has occurred

userLeasePeople –

Use:

The userLeasePeople table connects many people to many leases. When a landlord adds people to the lease agreements, that connection will be stored in this table.

Attributes:

Name	Type	Primary Key	Allow Nulls
leaseID	int	☑	☐
personID	int	☑	☐
role	varchar(50)	☐	☐

Relationships:

The following relationships exist for the userLeasePeople table:

Foreign Table	Foreign Column	Column	Type	Description
userLeases	leaseID	leaseID	Many-to-One	Many people on the lease can be associated to one lease
userPeople	personID	personID	Many-to-One	There can be many leases a given person is connected to

userLeaseFees –

Use:

The userLeaseFees table stores fees associated with leases and their occurrences, activation period requirements. It's a periodic fee template for the lease. A landlord enters the details about the lease fees when creating the lease. This information is pulled when the tenant calculates their due payments each month and when the landlord enters payment details.

Attributes:

Name	Type	Primary Key	Allow Nulls
leaseFeeID	int	☑	☐
leaseID	int	☐	☐
feeID	int	☐	☐
feeName	varchar(30)	☐	☐
feeAmount	float	☐	☐
occurrence	int	☐	☐
startAfterLength	int	☐	☐
startAfterPeriod	int	☐	☐
createUser	int	☐	☐
createDate	datetime	☐	☐

Relationships:

The following relationships exist for the userLeases table:

Foreign Table	Foreign Column	Column	Type	Description
userLeases	leaseID	leaseID	Many-to-One	Many lease fees can be associated to a given lease
appFeeTypes	feeID	feeID	Many-to-One	Many lease fees can use a single fee type
appOccurrences	occurrenceID	occurrence	Many-to-One	Many lease fees can use a single occurrence
appPeriod	periodID	startAfterPeriod	Many-to-One	Many lease fees will use one period item to determine if it's after it
userAccounts	userID	createUser	Many-to-One	Many lease fees can be created by the same user
userPaymentItems	leaseFeeID	leaseFeeID	One-to-Many	One lease fee can be applied to many payment items

appFeeTypes –

Use:

The appFeeTypes table stores categories of fees a landlord can apply to their lease. It includes a default value that cannot be changed by any user since it is an app table. A landlord selects from the fee type when creating the lease.

Attributes:

Name	Type	Primary Key	Allow Nulls
feeID	int	☑	☐
feeName	text	☐	☐
description	text	☐	☐
defaultPrice	decimal(8,2)	☐	☑
defaultOccurrence	int	☐	☑

Relationships:

The following relationships exist for the appFeeTypes table:

Foreign Table	Foreign Column	Column	Type	Description
userLeaseFees	feeID	feeID	One-to-Many	One fee type can be associated with many lease fees
appOccurrences	occurrenceID	defaultOccurrence	Many-to-One	Many fee types can use a single occurrence

appOccurrences –

Use:

The appOccurrences table stores occurrences of fees. It includes a occurrence value and a period over which the number of occurrences happen. For example, a landlord selects for rent to happen 1 occurrence per month or week or year.

Attributes:

Name	Type	Primary Key	Allow Nulls
occurrenceID	int	☑	☐
occurrence	int	☐	☐
perPeriod	int	☐	☑

Relationships:

The following relationships exist for the appOccurrences table:

Foreign Table	Foreign Column	Column	Type	Description
userLeases	leaseOccurrence	occurrenceID	One-to-Many	One occurrence can be used for many leases
userLeases	leaseSuccessionOccurrence	occurrenceID	One-to-Many	One occurrence can be used to dictate the behavior after the primary lease has ended on many leases
appFeeTypes	defaultOccurrence	occurrenceID	One-to-Many	One occurrence can be linked to many fee types
userLeaseFees	startAfterPeriod	occurrenceID	One-to-Many	One occurrence can be used for many lease fee start after periods
appPeriods	periodID	perPeriod	Many-to-One	Many occurrences can use a single period

appPeriods –

Use:

The appPeriods table stores durations that occurrences and income can be compared to. It includes a period name and its corresponding abbreviation. A landlord would use it when setting up a lease to dictate the frequency a fee (such as rent) is charged. A prospective tenant would use it to specify the frequency of their pay amount in their rental application.

Attributes:

Name	Type	Primary Key	Allow Nulls
periodID	int	☑	☐
name	varchar(20)	☐	☐
abbreviation	varchar(8)	☐	☐

Relationships:

The following relationships exist for the appPeriods table:

Foreign Table	Foreign Column	Column	Type	Description
appOccurrence	perPeriod	periodID	One-to-Many	One period could be used to specify many occurrences
userLeaseFees	startAfterPeriod	periodID	One-to-Many	One period will be used to specify the start time of many lease fees
userEmployment History	salaryPeriod	periodID	One-to-Many	One period can be used to describe the frequency of the employment's salary pay

userPaymentItems –

Use:

The userPaymentItems table records lease fee items that a user pays for during each payment. A landlord would record payment items when they log a payment from the tenant. A tenant would see the breakdown of items their payment covered on their invoice.

Attributes:

Name	Type	Primary Key	Allow Nulls
paymentItemID	int	☑	☐
paymentID	int	☐	☐
dueDate	date	☐	☐
itemName	varchar(50)	☐	☐
leaseFeeID	int	☐	☐
amountPaid	float	☐	☑
createUser	int	☐	☐
createDate	datetime	☐	☐

Relationships:

The following relationships exist for the userPaymentItems table:

Foreign Table	Foreign Column	Column	Type	Description
userPayments	paymentID	paymentID	Many-to-One	Many payment items can be connected to one payment
userAccounts	userID	createUser	Many-to-One	Many payment items can be created by one user
userLeaseFees	leaseFeeID	leaseFeeID	Many-to-One	Many payment items can reference the same lease fee (template)

userPayments –

Use:

The userPayments table records payments made by a tenant. A landlord would log a payment when it's received from the tenant. A tenant would see a history of their payments with an option to create a PDF invoice with the data.

Attributes:

Name	Type	Primary Key	Allow Nulls
paymentID	int	☑	☐
dueDate	date	☐	☐
paymentStatus	int	☐	☑
paymentMethod	int	☐	☑
amountReceived	float	☐	☑
dateReceived	date	☐	☐
createUser	int	☐	☐
createDate	datetime	☐	☐

Relationships:

The following relationships exist for the userPayments table:

Foreign Table	Foreign Column	Column	Type	Description
userPaymentItems	paymentID	paymentID	One-to-Many	One payment can be connected to many payment items
userAccounts	userID	createUser	Many-to-One	Many payments can be created by one user
appPaymentStatus	statusID	paymentStatus	Many-to-One	Many payments can have one status
appPaymentMethods	methodID	paymentMethod	Many-to-One	Many payments can use the same payment method

appPaymentStatus –

Use:

The appPaymentStatus table stores a list of payment status options. When a landlord logs a payment they will set the status of the payment. A tenant would see the status of their payments.

Attributes:

Name	Type	Primary Key	Allow Nulls	Default Value
statusID	int	☑	☐	
statusName	varchar(25)	☐	☐	
isCompleted	bit	☐	☐	((0))

Relationships:

The following relationships exist for the appPaymentStatus table:

Foreign Table	Foreign Column	Column	Type	Description
userPayments	paymentStatus	statusID	One-to-Many	One status can be linked to many payments

appPaymentMethods –

Use:

The appPaymentMethods table stores a list of payment method options. When a landlord logs a payment they will set the method of payment the tenant used.

Attributes:

Relationships:

The following relationships exist for the appPaymentMethods table:

Name	Type	Primary Key	Allow Nulls	I
methodID	int	☑	☐	
methodName	varchar(30)	☐	☐	

Foreign Table	Foreign Column	Column	Type	Description
userPayments	paymentMethod	methodID	One-to-Many	One payment method can be linked to many payments

userAccounts –

Use:

The userAccounts table stores information relating to the user's account credentials. When a landlord signs up for the service, they must create an account. When a tenant is setup with a lease on the system, they must have an account. If they want to use their account to auto-fill in an application, they must first login.

Attributes:

Name	Type	Primary Key	Allow Nulls	Default Value
userID	int	☑	☐	
accountTypeID	int	☐	☐	
emailAddress	int	☐	☐	
emailVerified	bit	☐	☐	((0))
passHash	varchar(50)	☐	☐	
salt	varchar(50)	☐	☑	
createDate	date	☐	☐	
attemptsSinceLastLogin	int	☐	☐	((0))

Relationships:

The following relationships exist for the userAccounts table:

Foreign Table	Foreign Column	Column	Type	Description
appAccountTypes	accountTypeID	accountTypeID	Many-to-One	Many accounts can use the same account type
userPeople	personID	userID	One-to-One	One account can be linked to one person
userPayments	createUser	userID	One-to-Many	One user can create many payments
userSessions	userID	userID	One-to-Many	One user can create many sessions
userProperties	createUser	userID	One-to-Many	One user can create many properties
userLease	createUser	userID	One-to-Many	One user can create many leases
userPaymentItems	createUser	userID	One-to-Many	One user can create many payment items
userLeaseFees	createUser	userID	One-to-Many	One user can create many lease fees
userCompanyRoles	userID	userID	One-to-Many	One user can hold many company-roles
userCompanyRoles	assignedUser	userID	One-to-Many	One user can assign many company-roles
userCompanyRoles	endedUser	userID	One-to-Many	One user can end many company roles
userCompanies	createdBy	userID	One-to-Many	One user can create many companies

appAccountTypes –

Use:

The appAccountTypes table stores account type options. A landlord has a given account type that gives them privileges to edit properties, leases, etc. A tenant has a different account type that restricts them from viewing certain information.

Attributes:

Name	Type	Primary Key	Allow Nulls
accountTypeID	int	☑	☐
typeName	varchar(40)	☐	☐
typeDescription	varchar(100)	☐	☑

Relationships:

The following relationships exist for the appAccountTypes table:

Foreign Table	Foreign Column	Column	Type	Description
userAccounts	accountTypeID	accountTypeID	One-to-Many	One account type can be used for many accounts
userCompanyRoles	roleID	accountTypeID	One-to-Many	One account type can be used for many company-roles

userSessions –

Use:

The userSessions table stores information surrounding an authenticated user session. When any user authenticates, the user session is stored in the database so any data request to the server can be verified.

Attributes:

Name	Type	Primary Key	Allow Nulls
sessionID	int	☑	☐
userID	int	☐	☐
loginDatetime	datetime	☐	☐
expiredDatetime	datetime	☐	☐
nextDatetime	datetime	☐	☐

Relationships:

The following relationships exist for the userSessions table:

Foreign Table	Foreign Column	Column	Type	Description
userAccounts	userID	userID	Many-to-One	Many sessions can be made for one user

userApplications – STRETCH GOAL

Use:

The userApplications table stores information contained in a perspective tenant's lease application. A potential tenant completes the lease application and the landlord reviews it and can make some comments relating to the rent-worthiness of the candidate.

Attributes:

Name	Type	Primary Key	Allow Nulls	Default Value
applicationID	int	☑	☐	
leaseID	int	☐	☐	
applicantID	int	☐	☐	
coApplicantID	int	☐	☑	
currentAddressID	int	☐	☐	
currentMonthlyRent	varchar(15)	☐	☐	
currentMoveIn	date	☐	☐	
currentLeaveReason	varchar(120)	☐	☑	
currentOwnerID	int	☐	☐	
previousAddressID	int	☐	☑	
previousMonthlyRent	varchar(15)	☐	☑	
previousMoveIn	date	☐	☑	
previousLeaveReason	varchar(120)	☐	☑	
previousOwnerID	int	☐	☑	
lastYearLatePayments	bit	☐	☐	
refusedPayments	bit	☐	☐	
everEvicted	bit	☐	☐	
additionalInfo	varchar(200)	☐	☑	
bestDayPhoneNo	varchar(20)	☐	☑	
bestEveningPhoneNo	varchar(20)	☐	☑	
informationReleaseID	int	☐	☐	
agree	bit	☐	☐	
dateTime	datetime	☐	☐	
ipAddress	varchar(15)	☐	☐	
applicationStatus	int	☐	☐	((0))
landlordComments	varchar(500)	☐	☑	

Relationships:

The following relationships exist for the userApplications table:

Foreign Table	Foreign Column	Column	Type	Description
userDependants	applicationID	applicationID	One-to-Many	One application may have many dependants
userVehicles	applicationID	applicationID	One-to-Many	One application may have many vehicles
userReferences	applicationID	applicationID	One-to-Many	One application may have many references
userEmploymentHistory	applicationID	applicationID	One-to-Many	One application may have many employment history entries
userLeases	leaseID	leaseID	Many-to-One	Many applications may exist for one lease
userPeople	personID	applicantID	Many-to-One	Many applications may have been made by one person
userPeople	personID	coApplicantID	Many-to-One	Many application may have one person as a co-applicant
userAddresses	addressID	currentAddressID	Many-to-One	Many applications may have the same applicant address
userPeople	personID	currentOwnerID	Many-to-One	Many applications may be made with the same current owner
userAddresses	addressID	previousAddressID	Many-to-One	Many applications may have the same previous applicant address
userPeople	personID	previousOwnerID	Many-to-One	Many applications may be made with the same previous owner

userDependants – STRETCH GOAL

Use:

The userDependants table stores information for people who are dependants on a person who holds the lease. A potential tenant specifies those who are dependants on the lease when they apply. Theoretically a landlord could charge a fee based on the number of dependants above a certain age.

Attributes:

Name	Type	Primary Key	Allow Nulls	Default Value
dependantID	int	☑	☐	
applicationID	int	☐	☑	
leaseID	int	☐	☑	
order	int	☐	☐	((1))
dependantName	varchar(50)	☐	☐	
dependantDOB	date	☐	☐	
applicantsRelationship	varchar(50)	☐	☑	

Relationships:

The following relationships exist for the userDependants table:

Foreign Table	Foreign Column	Column	Type	Description
userApplications	applicationID	applicationID	Many-to-One	Many dependants can be on one application
userLeases	leaseID	leaseID	Many-to-One	Many dependants can be on one lease

userEmploymentHistory – STRETCH GOAL

Use:

The userEmploymentHistory table stores information about a potential tenant's employment history. When a potential tenant applies for a lease, they need to disclose some information about their employment history including information about their supervisor and salary.

Attributes:

Name	Type	Primary Key	Allow Nulls
employmentID	int	☑	☐
applicationID	int	☐	☐
applicantID	int	☐	☑
employmentStatus	int	☐	☐
employerName	varchar(50)	☐	☐
employmentBegin	date	☐	☐
employedAs	varchar(50)	☐	☐
supervisorID	int	☐	☐
salary	float	☐	☐
salaryPeriod	int	☐	☐
landlordReviewUser	int	☐	☑
landlordComment	int	☐	☑

Relationships:

The following relationships exist for the userEmploymentHistory table:

Foreign Table	Foreign Column	Column	Type	Description
userApplications	applicationID	applicationID	Many-to-One	Many employment history records can be linked to one application
userPeople	personID	applicantID	Many-to-One	Many employment history records can be linked to the same applicant
userPeople	personID	supervisorID	Many-to-One	Many employment history records can have the same supervisor
appPeriods	periodID	salaryPeriod	Many-to-One	The frequency of many employment salary pay details can be linked to one period

userReferences – STRETCH GOAL

Use:

The userReferences table stores information about a potential tenant's references. When a potential tenant applies for a lease, they need to give multiple references that can support their lease-eligibility claim. The landlord can make remarks and flag if they've had communication with the reference.

Attributes:

Name	Type	Primary Key	Allow Nulls	Default Value
referenceID	int	☑	☐	
applicationID	int	☐	☐	
sortOrder	int	☐	☐	((1))
personID	int	☐	☐	
relationship	varchar(20)	☐	☐	
landlordReviewed	bit	☐	☐	((0))
landlordComments	varchar(500)	☐	☑	

Relationships:

The following relationships exist for the userReferences table:

Foreign Table	Foreign Column	Column	Type	Description
userApplications	applicationID	applicationID	Many-to-One	Many references can be linked to one application
userPeople	personID	personID	Many-to-One	Many references can be linked to the same person

userVehicles – STRETCH GOAL

Use:

The userVehicles table stores information about a potential tenant's vehicles. When a potential tenant applies for a lease, they need to list all vehicles that will be stored on the premises.

Attributes:

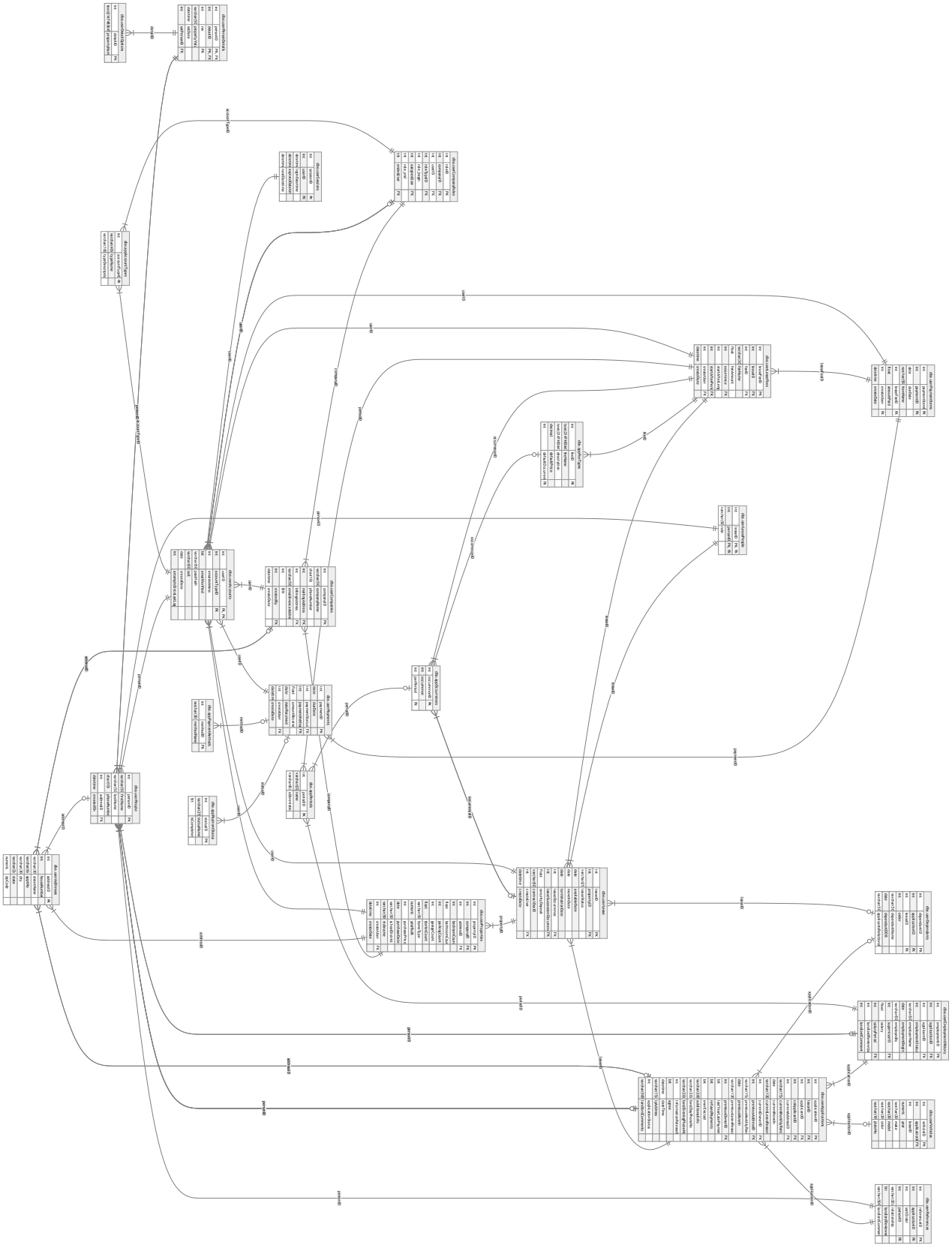
Name	Type	Primary Key	Allow Nulls
vehicleID	int	☑	☐
applicationID	int	☐	☑
leaseID	int	☐	☑
year	numeric(4,0)	☐	☑
make	varchar(20)	☐	☑
model	varchar(20)	☐	☑
color	varchar(20)	☐	☑
plateNo	varchar(10)	☐	☑

Relationships:

The following relationships exist for the userVehicles table:

Foreign Table	Foreign Column	Column	Type	Description
userApplications	applicationID	applicationID	Many-to-One	Many vehicles can be linked to one application

ER Diagram:



Revision Log:

Rev	Date	Modified By:	Changes
1	2023.11.12	Trevin	1. Documented relationships with userApplications table to the following tables as marked in red boxes: userLeases, userPeople, userAddresses, userPeople 2. Added userCompanyRoles table as marked in a red box 3. Updated the ER Diagram page
2	2023.11.12	Trevin	Marked DB tables that pertain to stretch goals as such
3	2023.11.26	Trevin	Made the createdBy field on the userCompanies table a Foreign Key from the userAccounts table {Marked with blue box}