Roman Krajewski - 553506
Leonid Ricker - 554903
Nils Gawlik - 553449

Dokumentation

Datenbank für Brettspielverleih

Krajewski, Gawlik, Ricker

18. Januar 2017

Inhaltsverzeichnis

Kurzbeschreibung

Ein Team aus fünf Brettspiel-Enthusiasten entwickelt einen Online-Brettspielverleih. Hierfür soll eine Datenbank konzipiert werden.
Diese soll zunächst Kundendaten verwalten und einen Überblick über das Brettspielinventar bieten. Des weiteren wird gespeichert, welcher Nutzer welches Spiel über welchen Zeitraum ausleiht.

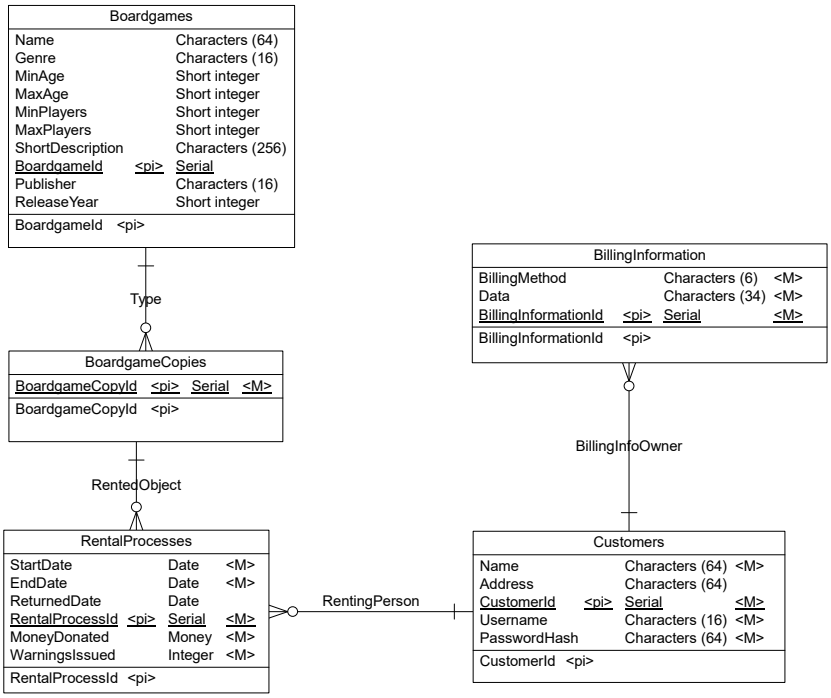
Allgemeiner Funktionsumfang

Die Datenbank verwaltet Daten rund um das Ausleihen von Brettspielen.
Es werden Prozeduren zum eintragen und löschen von Nutzern und Spielen sowie zum ändern dieser Daten bereitgestellt. Von Brettspielen werden sowohl Informationen zu den verschiedenen vorhandenen Spielen gespeichert als auch einzelne Kopien dieser Spiele erfasst.
Man kann jederzeit feststellen, welche Brettspielkopien zu welcher Zeit von welchen Nutzer ausgeliehen waren oder immer noch ausgeliehen sind. Auch kann man sehen welche Nutzer wegen Überziehung zu mahnen sind und nach erfolgreicher Mahnung die Ausleihvorgänge als "gemahnt" setzen.
Des Weiteren gibt es noch statistische Features, wie z.B. die spendabelsten Nutzer oder die populärsten Spiele anzuzeigen.

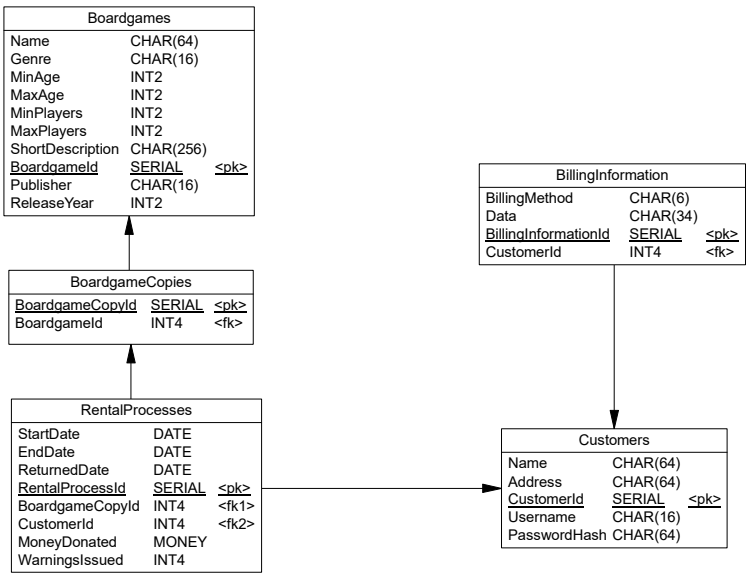
Technik und Software

Als Server benutzen wir für unser Projekt den Projekteserver des Fachbereichs 4 (https://studi.f4.htw-berlin.de/www/).
Wir haben den Sybase PowerDesigner für die Erstellung unseres ERMs verwendet und diesen ein Script für die Erstellung der Tabellen in einer Datenbank generieren lassen.
Als DBMS benutzen wir PostgreSQL. Für die Administration phpPgAdmin und DataGrip. PostgreSQL und phpPgAdmin sind auf dem Projekteserver vorinstalliert, was uns eine stabile und zuverlässige Umgebung gibt um zu dritt an der Datenbank zu arbeiten.
Zur Generierung von plausiblen Testdaten benutzen wir Mockarroo (mockaroo.com), kombiniert mit JavaScript zur Extraktion von Brettspiel-Titeln aus Wikipedia.

Konzeptuelles Datenmodell



Physisches Datenmodell



Implementierungshinweise

Voraussetzung für die Implementierung unsere Datenbank ist ein Server, auf dem Postgresql installiert ist. Die Tabellen müssen zuerst erstellt werden, da unsere Views von diesen abhängig sind. Dazu muss das Script crebas.sql ausgeführt werden. Anschließend können mit dem Script viewsfunctions.sql die Views und Funktionen hinzugefügt werden. Die Scripte testDataBig.sql und testDataSmall.sql fügen Testdaten in die Datenbank ein, die bei Bedarf verwendet werden können.

Funktionen

Name	**gamesAvailable**
Anwendungszweck	Gibt die Anzahl an momentan nicht ausgeliehenen Kopien des Spiels mit der spezifizierten gameId zurück.
Parameter	gameId INT
Rückgabewert	BIGINT

Name	**rentGame**
Anwendungszweck	Leiht ein Spiel mit der spezifizierten Game-Id für den spezifizierten Kunden. Des Weiteren anzugeben sind Ausleihdauer, die Höhe des gezahlten Solidarbeitrags. Wenn nicht anders angegeben is das Ausleihdatum der heutige Tag, es können aber auch rückwirkend Spiele ausgeliehen werden indem dieses spezifiziert wird. Es wird zurückgegeben ob der Ausleihvorgang erfolgreich war.
Parameter	aCustomerId INT, aGameId INT,aMoneyDonated NUMERIC, aRentedDays INT [, aStartDate] DATE
Rückgabewert	BOOL

Name	**mostRentedByPlayer**
Anwendungszweck	Zeigt, welches Spiel ein bestimmter Spieler am häufigsten ausgeliehen hat.
Parameter	customerid INT
Rückgabewert	gameid INT

Name	**customersThatRentedGame**
Anwendungszweck	Zeigt die Spieler, die ein Spiel mindestens so oft ausgeliehen haben, wie als Parameter eingegeben.
Parameter	aGameId INT, aMinTimesRented INT
Rückgabewert	Table(customerId INT, timesRented BIGINT)

Prozeduren

In Postgresql gibt es keinen syntaktischen Unterschied zwischen Funktionen und Prozeduren. Wir haben deshalb Funktionen ohne Rückgabewert haben. Diese könnte man semantisch als Prozeduren bezeichnen.

Name	**increaseWarnings**
Anwendungszweck	Erhöht die Anzahl der ausgestellten Warnungen für den spezifizierten Ausleihprozess um die spezifizierte Anzahl. Diese Funktion sollte nur dann aufgerufen werden wenn ein externes Programm tatsächlich eine Warnung versendet hat (per e-mail, Post, etc.). Um die überfälligen Ausleihprozess-Ids zu erhalten sollte die Sicht WarningsNecessary verwendet werden.
Parameter	aRentalProcessId INT, aWarnings INT
Rückgabewert	-

Name	**changeCustomerAddress**
Anwendungszweck	Ändert die Adresse des customers mit der angegebenen Id.
Parameter	aCustomerId INT, aNewAddress CHAR(64)
Rückgabewert	-

Views

Name	**customerWithMostRentals**
Anwendungszweck	Enthält die Spieler, die am meisten Spiele ausgeliehen haben.
Spalten	customerId, numberOfRentals

Name	**activeRentals**
Anwendungszweck	Enthält die Ausleihen, die noch nicht abgegeben wurden.
Spalten	RentalProcesses.*

Name	**availableGameCopies**
Anwendungszweck	Enthält die Brettspiel-Kopien, die zurzeit ausleihbar sind.
Spalten	BoardgameCopyId INT, BoardgameId INT

Name	**warningsNecessary**
Anwendungszweck	Enthält die Ausleihprozesse, für die noch Mahnungen ausstehen. Diese Sicht sollte von einem externen Programm benutzt werden, dass diese Mahnungen versendet. Mahnungen können anschließend mit der Prozedur increaseWarnings bestätigt werden.
Spalten	rentalprocessid INT, boardgamecopyid INT , daysoverdue INT, warningsnecessary INT, name CHAR(64), address CHAR(64)

Name	**mostProfitableGames**
Anwendungszweck	Enthält die Spiele, die am für die insgesamt am meisten gezahlt wurde.
Spalten	name CHAR(64), boardgameid INT, moneySum MONEY

Name	**mostRentedGames**
Anwendungszweck	Enthält die Spiele, die insgesamt am häufigsten ausgeliehen wurden.
Spalten	name CHAR(64), boardgameid INT, timesRented INT

Views fortgesetzt auf nächster Seite...

Name	**mostRentedByUniquePlayers**
Anwendungszweck	Enthält die Spiele, die insgesamt von den meisten verschiedenen Spielern ausgeliehen wurden.
Spalten	name CHAR(64), boardgameid INT, timesRented INT

Verbesserungs- und Erweiterungsmöglichkeiten

Um von einem Endanwender bedient zu werden fehlt unserer Datenbank noch ein Benutzerinterface. Außerdem könnte man den Server noch so verbessern, dass er automatisch Mahnungen generiert und versendet. Man könnte auch Zeiten anstatt Daten verwenden um die Ausleihen zu verwalten, was ermöglichen würde ein Spiel am selben Tag auszuleihen und zurückzubringen.

Quellen

Lehrunterlagen der des Moduls Datenbanken im Wintersemester 2016/17, Internationaler Studiengang Medieninformatik, HTW Berlin - Rüger Oßwald
Die PostgreSQL Dokumentation - https://www.postgresql.org/docs/
Präsentations-Folien zu Funktionen in PostgreSQL - http://www.joeconway.com/presentations/function_basics.pdf