Fahrschule

Meilenstein 1 (ursprüngliche Version): Anforderungsanalyse & Konzeptioneller Entwurf

Anforderungsanalyse:

Das **Fahrschulzentrum** (FZ) hat einen Name (der es eindeutig charakterisiert), eine PLZ und einen Ort. Das FZ umfasst mehrere **Gebäude**, wobei ein Gebäude nur zu einem FZ zugeordnet werden kann. Ein Gebäude hat eine Gebäudenummer (die es eindeutig charakterisiert), einen Name und eine Adresse. Ein Gebäude kann wiederum mehrere **Räume** haben, die jeweils nur einem Gebäude zugeordnet sind. Ein Raum hat eine Raumnummer (die aber nur innerhalb des Gebäudes eindeutig ist), einen Name und eine Einrichtung.
Das FZ beschäftigt mehrere **Mitarbeiter**, die jeweils nur dem einen FZ angehören. Mitarbeiter werden beschrieben durch eine (eindeutige) Personalnummer, einen Name, eine SVNR (= Sozialversicherungsnummer) und ein Gehalt. Ein Mitarbeiter kann entweder ein ***Admin***, der als weitere Eigenschaft eine Funktion hat, oder ein ***Fahrlehrer*** (mit einer Zulassungsnummer) sein. Ein Mitarbeiter arbeitet in einem Raum, wobei jeder Mitarbeiter allein diesen Raum zur Verfügung hat. Zwecks der Qualitätssicherung bewerten Fahrlehrer andere Fahrlehrer, wobei jeder Fahrlehrer mehrere seiner Kollegen bewerten kann und auch von mehreren bewertet werden kann.
Mehrere Admins und ein Fahrlehrer können zusammen mehrere **Fahrkurse** koordinieren. Ein Fahrkurs hat eine (eindeutige) Kursnummer, ein Tagesdatum, eine Beginnzeit und eine Endzeit.
Schließlich bucht ein **Kunde**, der eine (eindeutige) Kundennummer, einen Name und eine Reisepassnummer hat, beliebig viele Fahrkurse. Ein Fahrkurs kann nur von einem Kunden gebucht werden.

Konzeptioneller Entwurf (ER-Diagramm) (alt):

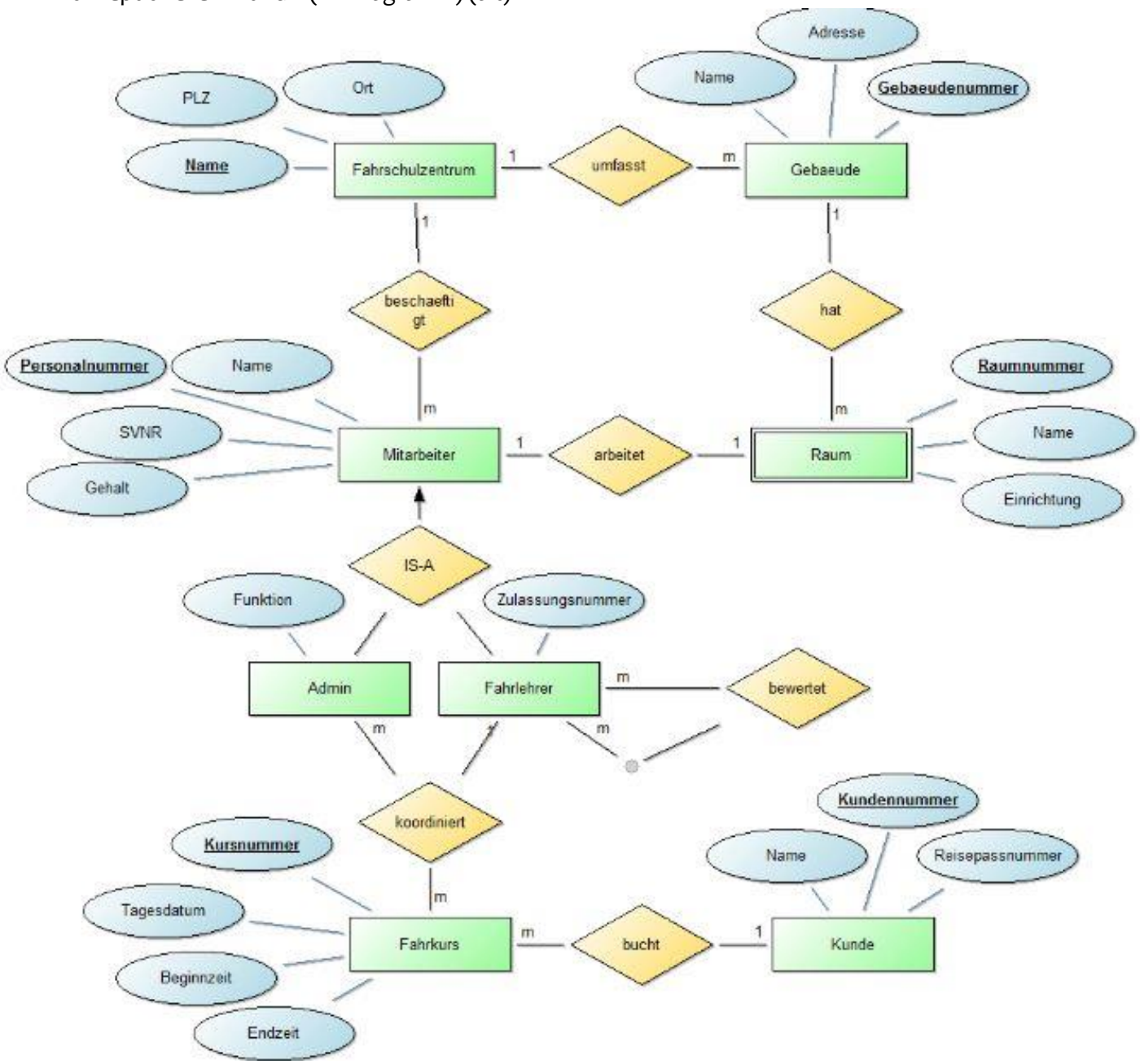


Abbildung 1: Fahrschule – Entity Relationship Diagramm (alt)

Meilenstein 1 (neu): Anforderungsanalyse & Konzeptioneller Entwurf

Anforderungsanalyse:

Das **Fahrschulzentrum** (FZ) hat eine FahrschulzentrumID (FZID) (die es eindeutig charakterisiert), einen Name, eine PLZ und einen Ort. Das FZ umfasst mehrere **Gebäude**, wobei ein Gebäude nur zu einem FZ zugeordnet werden kann. Ein Gebäude hat eine Gebäudenummer (die es eindeutig charakterisiert), einen Name und eine Adresse. Ein Gebäude kann wiederum mehrere **Arbeitszimmer** haben, die jeweils nur einem Gebäude zugeordnet sind. Ein Arbeitszimmer hat eine Raumnummer (die aber nur innerhalb des Gebäudes eindeutig ist), einen Name und eine Einrichtung.
Das FZ beschäftigt mehrere **Mitarbeiter**, die jeweils nur dem einen FZ angehören. Mitarbeiter werden beschrieben durch eine (eindeutige) Personalnummer, einen Name, eine SVNR (= Sozialversicherungsnummer) und ein Gehalt. Ein Mitarbeiter kann entweder ein ***Admin***, der als weitere Eigenschaften eine Funktion, eine Admin-ID und ein Hierarchielevel hat, oder ein ***Fahrlehrer*** (mit einer Zulassungsnummer, Fahrlehrer-ID und einer Spezialisierung) sein. Ein Mitarbeiter arbeitet in einem Raum, wobei jeder Mitarbeiter allein diesen Raum zur Verfügung hat.
Mehrere Admins und ein Fahrlehrer können zusammen mehrere **Fahrkurse** koordinieren. Ein Fahrkurs hat eine (eindeutige) Kursnummer, ein Tagesdatum, eine Beginnzeit und eine Endzeit.
Schließlich bucht ein **Kunde**, der eine (eindeutige) Kundennummer, einen Name und eine Reisepassnummer hat, beliebig viele Fahrkurse. Ein Fahrkurs kann nur von einem Kunden gebucht werden. Ein Kunde kann mehrere Kunden kennen (i.e. Kollege von ihnen sein), wobei jeder Kunde wieder mehrere seiner Kollegen kennen kann. Ein Kunde benützt individuell abgestimmt ein **Fahrzeug**, und ein Fahrzeug kann nur von einem Kunden benützt werden.

Konzeptioneller Entwurf (ER-Diagramm) (neu):

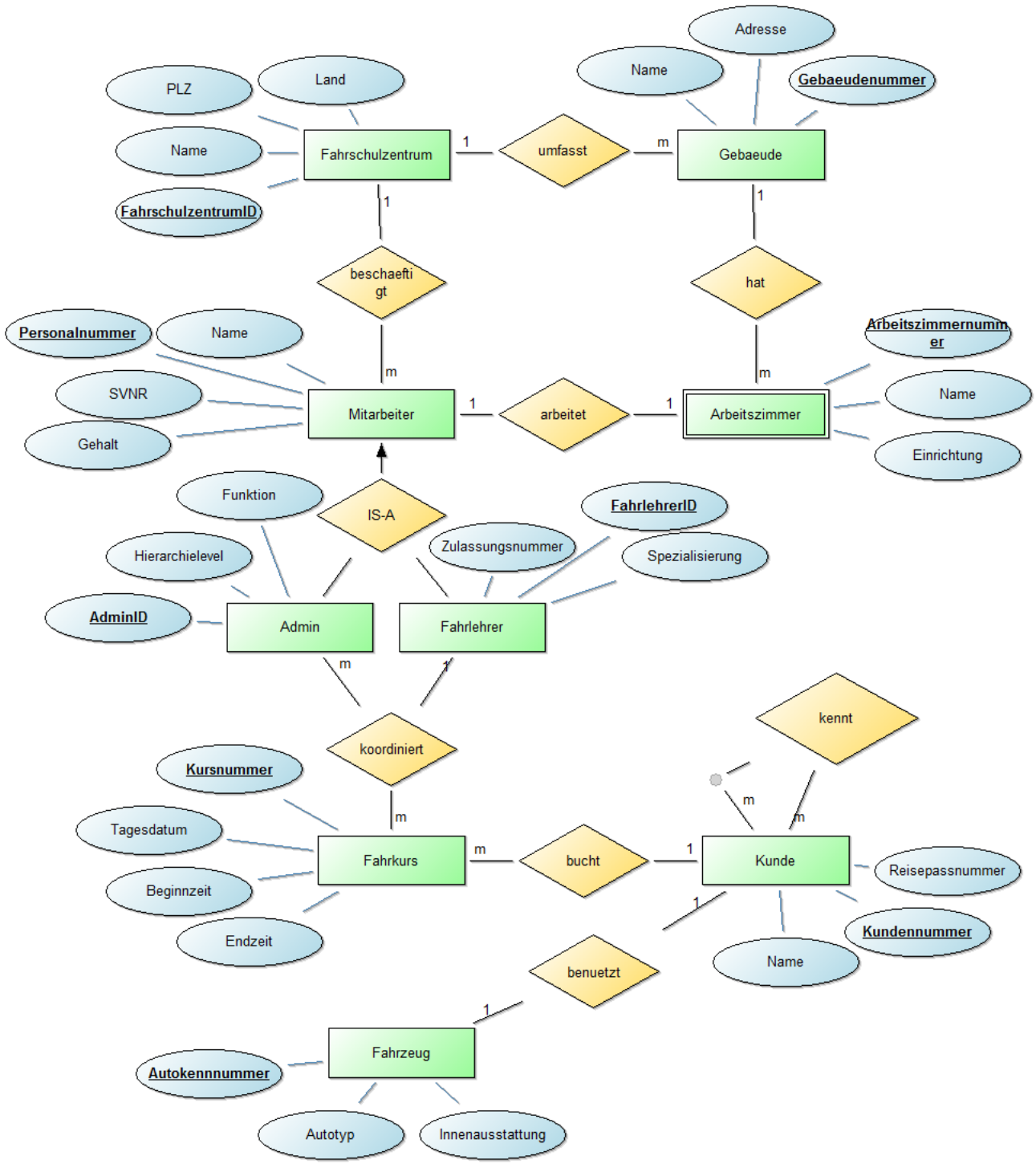


Abbildung 2: Fahrschule – Entity Relationship Diagramm (neu)

Anmerkung: Die Relation „koordiniert", die zu einer eigenen Tabelle wird, erhält die Attribute „KoordinationsID" (= künstlicher PK → löst den umfangreichen PK über Personalnr1+Personalnr2+AdminID+FahrlehrerID+Kursnr ab) und „Geheimreferenz" (= wird verwendet, wenn in der Öffentlichkeit über eine bestimmte interne Sachverhalte geredet wird).

Meilenstein 2: Logischer Entwurf

Relationenschemata zum ER-Diagramm „Fahrschule“:

Fahrschulzentrum (FZID, FZName, PLZ, Ort)
PK: FZID

Gebaeude (Gebaeudenr, GBName, Adresse, FZID)
PK: Gebaeudenr
FK: Gebaeude.FZID ◊ Fahrschulzentrum.FZID

Arbeitszimmer (Arbeitszimmernr, AZName, Einrichtung, *Gebaeudenr*, *Personalnr*)
PK: Arbeitszimmernr, *Gebaeudenr*
FK: Arbeitszimmer.Gebaeudenr ◊ Gebaeude.Gebaeudenr
FK: Arbeitszimmer.Personalnr ◊ Mitarbeiter.Personalnr

Mitarbeiter (Personalnr, MAName, SVNR, Gehalt,*FZID*)
PK: Personalnr
FK: Mitarbeiter.FZID ◊ Fahrschulzentrum.FZID

Admin (Personalnr, AdminID, MAName, SVNR, Gehalt, Hierarchielevel, Funktion)
PK: Personalnr, AdminID

Fahrlehrer (Personalnr, FahrlehrerID, MAName, SVNR, Gehalt, Zulassungsnr, Spezialisierung)
PK: Personalnr, FahrlehrerID

Koordination (KoordinationsID, *Personalnr1*, *FahrlehrerID*, *AdminID*, *Personalnr2*, *Kursnr*, Geheimreferenz)
PK: KoordinationsID (künstlicher PK, löst ab: *Personalnr1-Personalnr2-FahlehrerID-AdminID-Kursnr)*
FK: Koordination.Personalnr1 ◊ Admin.Personalnr
FK: Koordination.Personalnr2 ◊ Fahrlehrer.Personalnr
FK: Koordination.AdminID ◊ Admin.AdminID
FK: Koordination.FahrlehrerID ◊ Fahrlehrer.FahrlehrerID
FK: Koordination.Kursnr ◊ Fahrkurs.Kursnr
(Anmerkung: im physischen Entwurf wurden vereinfachend eine koordid (i.e. KoordinationsID) und eine Geheimreferenz eingeführt)

Fahrkurs (Kursnr, Tagesdatum, Beginnzeit, Endzeit,*Kundennr*)
PK: Kursnr
FK: Fahrkurs.Kundennr ◊ Kunde.Kundennr

Kunde (Kundennr, Reisepassnr, KName)
PK: Kundennr

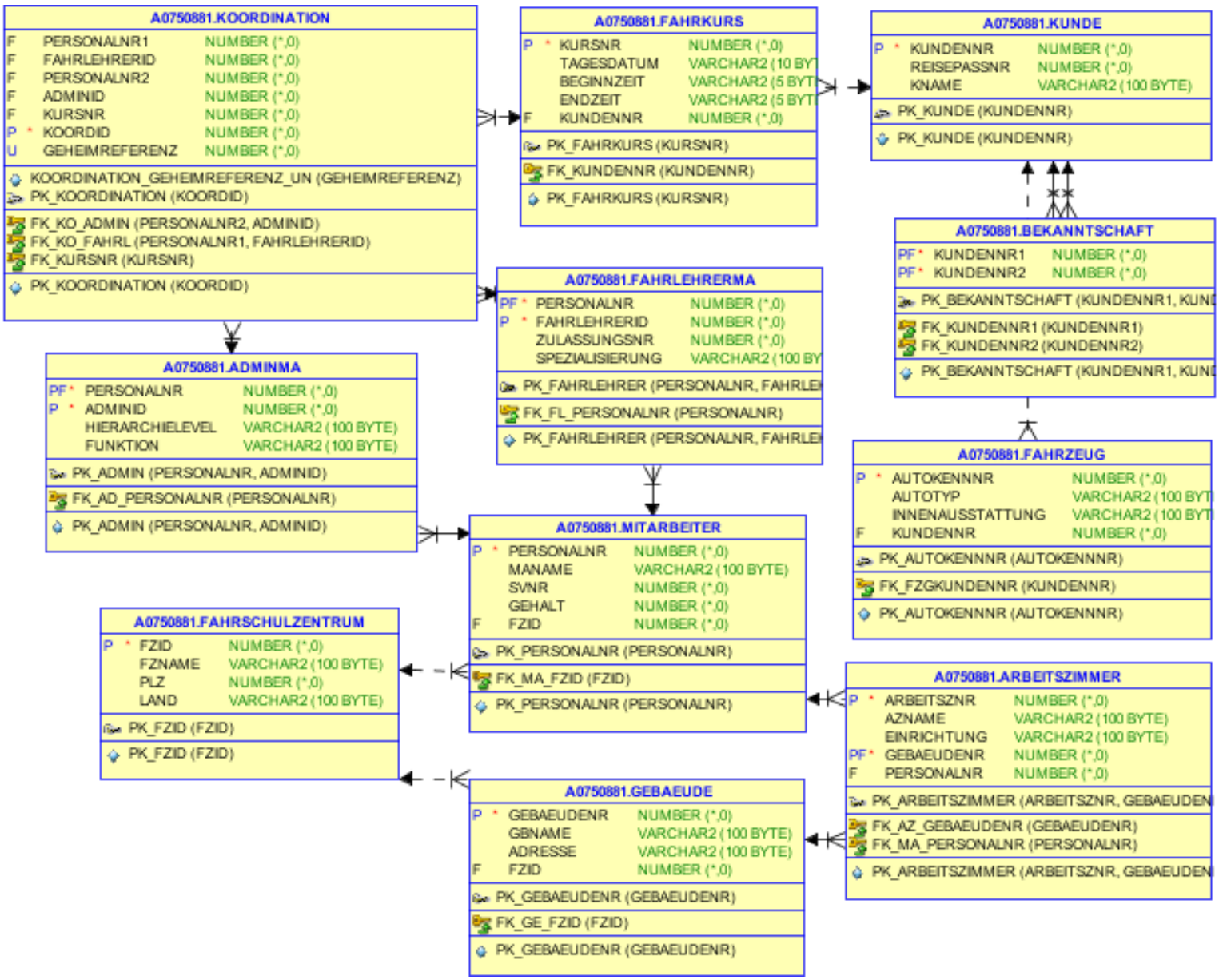
Bekanntschaft (*Kundennr1*, *Kundennr2*)
PK: *Kundennr1*, *Kundennr2*
FK: Bekanntschaft.Kundennr1 ◊ Kunde.Kundennr
FK: Bekanntschaft.Kundennr2 ◊ Kunde.Kundennr

Fahrzeug (Autokennnr, Autotyp, Innenausstattung, *Kundennr*)
PK: Autokennnr
FK: Fahrzeug.Kundennr ◊ Kunde.Kundennr

Meilenstein 3: Physischer Entwurf (SQL)

Siehe SQL-Files!

Relationales Modell (mittels SQL Developer erstellt):



Meilenstein 4+5: Implementierung

Java Implementierung

In das gegebene Java-File „**TestDataGenerator1**“ werden die angeforderten Hauptzielsetzungen (i.e. realistische Größenordnungen) entsprechend die Tabellen Kunde mit 2000 Einträgen (Insert-Statements), Fahrzeug mit 2000 Einträgen und Fahrschulzentrum mit 300 Einträgen befüllt.
Um die Daten realistisch darzustellen wurde zusätzlich das Java-File „**Datenliste**“, aus dem die Tabellen über die Insert-Statements Informationen beziehen (und zufällig zugeteilt bekommen, wo dies möglich und sinnvoll ist).
Danach werden jeweils die Einträge der Tabellen gezählt mittels „SELECT COUNT(*) FROM table“-Funktion und die jeweilige Anzahl der eingefügten Einträge ausgegeben.
Danach werden alle Sets, Statements und Connections geschlossen.
Im Java-File „**TestDataGeneratorRest**“ werden die weiteren Tabellen mit INSERT-Statements befüllt. Beim Fahrkurs werden die Spalten abgesehen von der „kursnr“ befüllt, da diese schon mittels auto_increment-Funktion im SQL erstellt wird. Auch für dieses INSERT-File wurde eine Datenliste („**DatenlisteRest**“) angelegt, um eine umfangreichere Auswahl an Daten für die INSERTS bereitzustellen. Die Datenzuteilung erfolgt wieder zufällig.

PHP Implementierung

Das Programm zeigt für 5 verschiedene Entitäten (lt. Angabe gefordert: 4) die Ausgabe eines SQL-Statements (hier: SELECT-Statements; jeweils SELECT*FROM entity xy und ein spezielles Statement) auf der Website. Es wird jeweils ein searchvalue spezifiziert, der als Parameter übergeben wird und mittels dem in der Suchmaske in den vorhandenen Dateneinträgen gesucht werden kann.

- Fahrschulzentrum: SELECT*FROM fahrschulzentrum WHERE fzid = searchvalue
- Gebaeude: SELECT*FROM gebaeude WHERE gbname = searchvalue
- Mitarbeiter: SELECT*FROM mitarbeiter WHERE maname = searchvalue
- Arbeitszimmer: SELECT*FROM arbeitszimmer WHERE azname = searchvalue
- Fahrkurs: SELECT*FROM fahrkurs WHERE beginnzeit = searchvalue

Der **Aufruf des Programms** erfolgt über:
http://wwwlab.cs.univie.ac.at/~a0750881/sql/meilenstein5/index.php .

Beispiel „Anzeige aller Einträge“: ***Untenstehend ein Screenshot als Ausschnitt der Anzeige aller (in diesem Projekt: 300) Einträge für die Entität FAHRSCHULZENTRUM:***

/ FAHRSCHULZENTRUM

Alle Fahrschulzentren (fzid) --- Suche nach :
Notice: Suchen!

fzid	fzname	plz	land
3000	FS Murach	1000	Oesterreich
3001	FS Beberg	1001	Oesterreich
3002	FS Linz01	1002	Oesterreich
3003	FS Linz18	1003	Oesterreich
3004	FS Mauern	1004	Oesterreich
3005	FS Graz06	1005	Oesterreich
3006	FS Dauern	1006	Oesterreich
3007	FS Wels05	1007	Oesterreich
3008	FS Jurach	1008	Oesterreich
3009	FS Wels11	1009	Oesterreich
3010	FS Graz15	1010	Oesterreich
3011	FS Wels04	1011	Oesterreich
3012	FS Graz05	1012	Oesterreich
3013	FS Ceberg	1013	Oesterreich
3014	FS Jauern	1014	Oesterreich
3015	FS Graz09	1015	Oesterreich
3016	FS Graz01	1016	Oesterreich
3017	FS Cauern	1017	Oesterreich

Beispiel „Anzeige selektiver Einträge“: *Untenstehend ein Screenshot der Anzeige aller (in diesem Projekt 300) Einträge für die Entität MITARBEITER (wenn man nach „Marlene“ sucht – es gibt 5 Mitarbeiter, die „Marlene“ heißen):*

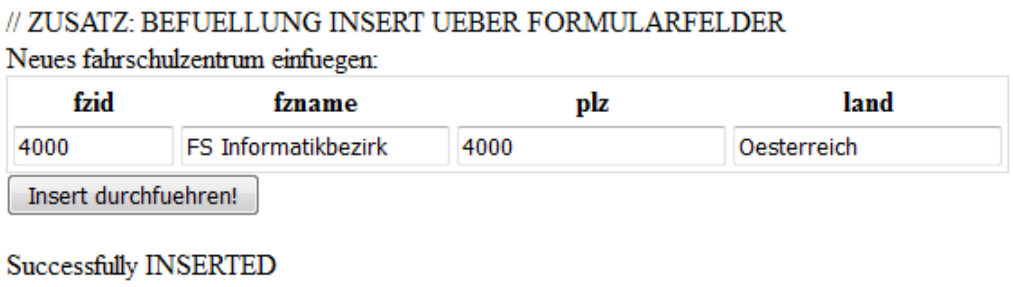
// MITARBEITER

Alle Mitarbeiter (maname) --- Suche nach : Marlene Suchen!

personalnr	maname	svnr	gehalt	fzid
1006	Marlene Cerger	700006	2000	3006
1007	Marlene Derger	700007	4000	3007
1008	Marlene Aerger	700008	1000	3008
1013	Marlene Berger	700013	2000	3013
1014	Marlene Ferger	700014	1000	3014

Insgesamt 5 mitarbeiter gefunden!

Beispiel „Befüllung eines Inserts über Formularfelder in der Eingabemaske“: Ein neues Fahrschulzentrum kann über die Eingabemaske mittels Feldern für „fzid“, „fzname“, „plz“ und „land“ eingefügt werden – wenn es funktioniert hat, wird „Successfully INSERTED“ angezeigt. Untenstehend der Screenshot eines erfolgreichen Inserts:



Wenn dann nach dem neu eingefügten Eintrag gesucht wird, wird dieser in der Liste für die Fahrschulzentren angezeigt (wenn man nach seiner „fzid“ über die Suchmaske sucht). Untenstehend ein Screenshot zum Beweis, dass der obige Eintrag nun existiert:

Beispiel „Erstellung von Stored Procedures“: Zunächst wurde die Stored Procedure im SQL angelegt (siehe SQL-Datei) und dann über PHP aufgerufen.
Auszug aus der SQL-Datei zur Erstellung der Stored Procedure:

```
-- STORED PROCEDURE ANLEGEN:
create or replace PROCEDURE personalnr_arbeitsznr(persnr IN NUMBER, aznr OUT NUMBER) IS
BEGIN
     Select a.arbeitsznr INTO aznr from mitarbeiter p, arbeitszimmer a
     where p.personalnr=persnr AND a.personalnr=persnr;
END;
```

Mittels dieser Stored Procedure wird die Personalnummer eines Mitarbeiters entgegengenommen (als Inputparameter) und geschaut, in welchem Arbeitszimmer (i.e. Outputparameter Arbeitszimmernummer) der angefragte Mitarbeiter arbeitet.

Die Anzeige auf der Website für den Mitarbeiter mit der angefragten Personalnummer sieht dann folgendermaßen aus (hier z.B. für die Mitarbeiter mit der Personalnummer 1002 und 1003):

// ZUSATZ: ERSTELLUNG EINER STORED PROCEDURE
Suche Arbeitszimmernummer zu bestimmtem Mitarbeiter (personalnr): 1002 Aufruf Stored Procedure!

1002 arbeitet in Arbeitszimmer mit Nummer 102

// ZUSATZ: ERSTELLUNG EINER STORED PROCEDURE
Suche Arbeitszimmernummer zu bestimmtem Mitarbeiter (personalnr): 1003 Aufruf Stored Procedure!

1003 arbeitet in Arbeitszimmer mit Nummer 103