ABDULLAH GÜL
ÜNİVERSİTESİ

Share Cloths
Comp204 Term Project
Phase 2

Instructor
Ahmet Soran
Samet Tonyalı

Group Members
Tacettin Batuhan Bostancı
Ayşe Şeyda Çalışkan
Mustafa Demiröz
Mehmet Anıl İrfanoğlu
Dhiya Ulhaq

TABLE OF CONTENTS

LIST OF FIGURES

Abstract

This article contains means of implementation of a charity database system. Charities have significant obligation of credibility. The aim of this project is to make certain of the credibility of the help foundation by acquiring trust from the donators. The application offers for the donators to see status of their donation and to the charities to get the reputation of being trustworthy. The general mechanism of the project is outlined in the ER diagram, database design, system requirements, tools used, analysis and specifications in detail. Lastly, some screenshots from the application are given in the paper for better understanding of the project.

SUMMARY

This report is based on the project that creates a database that has been established to ensure that the connection between people who want to give their clothes to help and those who need them is faster and more effective. In the realization of this project, SQL to create the connection between people and systems, Flutter for the creation of the website, and programs such as Lucid App and Draw.io as additional resources were utilized. In the following parts of the report, general description, requirement analysis, specifications, IDEs, UML, High-level diagram, E-R diagram, design philosophy, cardinalities, and user permission of the project are mentioned in detail.

REQUIREMENT ANALYSIS

The Share Clothes application have charity page that user will enter the needed information about the clothes that they gave.

The clothes will have different attributes such as cloth id, type, size, user id and hangar id. Till the cloth reaches the hangar, hangar id will be null.

Once user fill out the charity page program will create a transportation record and send a delivery personal to user's can bring those cloths to hangar by itself in that case there transporter id in the transportation record will be null.

The user is a person and in addition to user id, username and password it has person's attributes which are name, surname, sex and address id.

The Gathering hangars that these cloths will be kept haves three attributes hangar id, hangar name and address id. Then these cloths will be carried out to predetermined families and places and distributed there.

People who got aided will have person id, upper size, lower size as body size and the number of clothes that he/she got aided.

The user will be able to track down the status of the cloth by entering id of the cloth.

The application will offer list of clothes in specified gathering hangar to the charity organization so that they could list by specified size and type.

SPECIFICATION

As we know, the need for clothing is an important part of every person's life. Many organizations provide clothing aid to poor regions where this need cannot be met. After long observations, although we saw that these charities did not keep records to keep the donation confidential, we came to the conclusion that this actually caused many problems. The reason for this is that the clothes given go to other places rather than the actual destination. Therefore, we decided to set up a database. Thanks to this system, users will open an account and donate clothes and give details about the clothes they give. The charity employees collecting these clothes will make a classification process based on the characteristics of the dress givers entered into the system. Finally, people who need clothes will register to the system and give some information. To give an example of this information, these are some elements that are important in giving clothes such as height, weight, body size. When all these things are implemented, a very effective clothe sharing system emerges. and thanks to these records, curious people will be able to confirm that the dresses are reaching the people who really need it.

GENERAL DESCRIPTION

The cloth aid system our purpose is to make sufficient and organizable application with the MySQL database management system. In the system, there are 2 significant data one of them is people who need clothes and the other one is people who sent his/him clothes. In the cloth aid system, we will keep the recipient, donor information. Determining the size of the clothes that will go to help beforehand, understanding who needs which clothes by the system, we will be late for the wrong clothes to go. At the same time, we aim to make the aid campaign more contractual and efficient by keeping the information of the people who help and the people helped, the information of the transfer center, and the information of the people who will help. In this way, by having a general background by the charity organization, aid campaign information can be easily sent to the same people in future aid campaigns.

TOOLS/IDES

MySQL Workbench:
It is the most important program in order to complete Project. It will be used to create and manage database system of the Project.

NetBeans:
This application is used to create desktop application. Using NetBeans is much more easier than using eclipse because there is no help while creating user interface in eclipse but in NetBeans, Java SWING has lots of features that can be used in projects.

Java:
Java is used in NetBeans in order to create projects which works with MySQL. As we know Java is very detailed language and it offers lots of important features which are necessary to create very good project.

GitHub:
It will be used to share the versions of mobile application.

HIGH LEVEL DIAGRAM

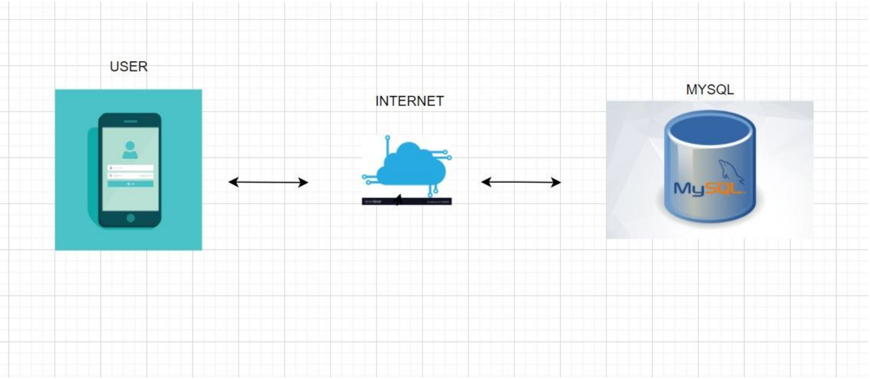


Figure1.: High Level Diagram

UML USE CASE DIAGRAM

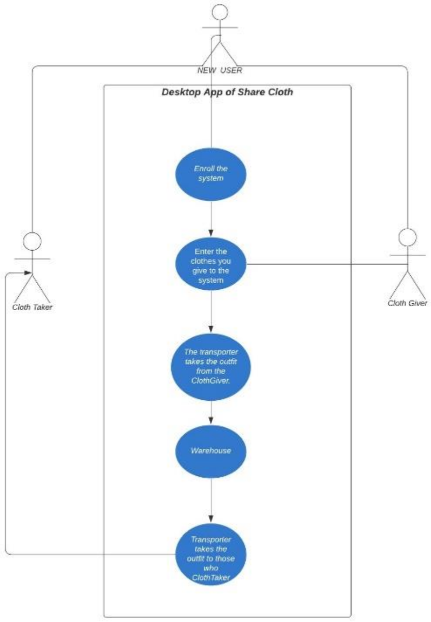

NEW USER
Desktop App of Share Cloth
Enroll the system
Enter the clothes you give to the system
The transporter takes the outfit from the ClothGiver.
Warehouse
Transporter takes the outfit to those who ClothTaker
Cloth Taker
Cloth Giver

Figure2.: UML Diagram

E-R DIAGRAM

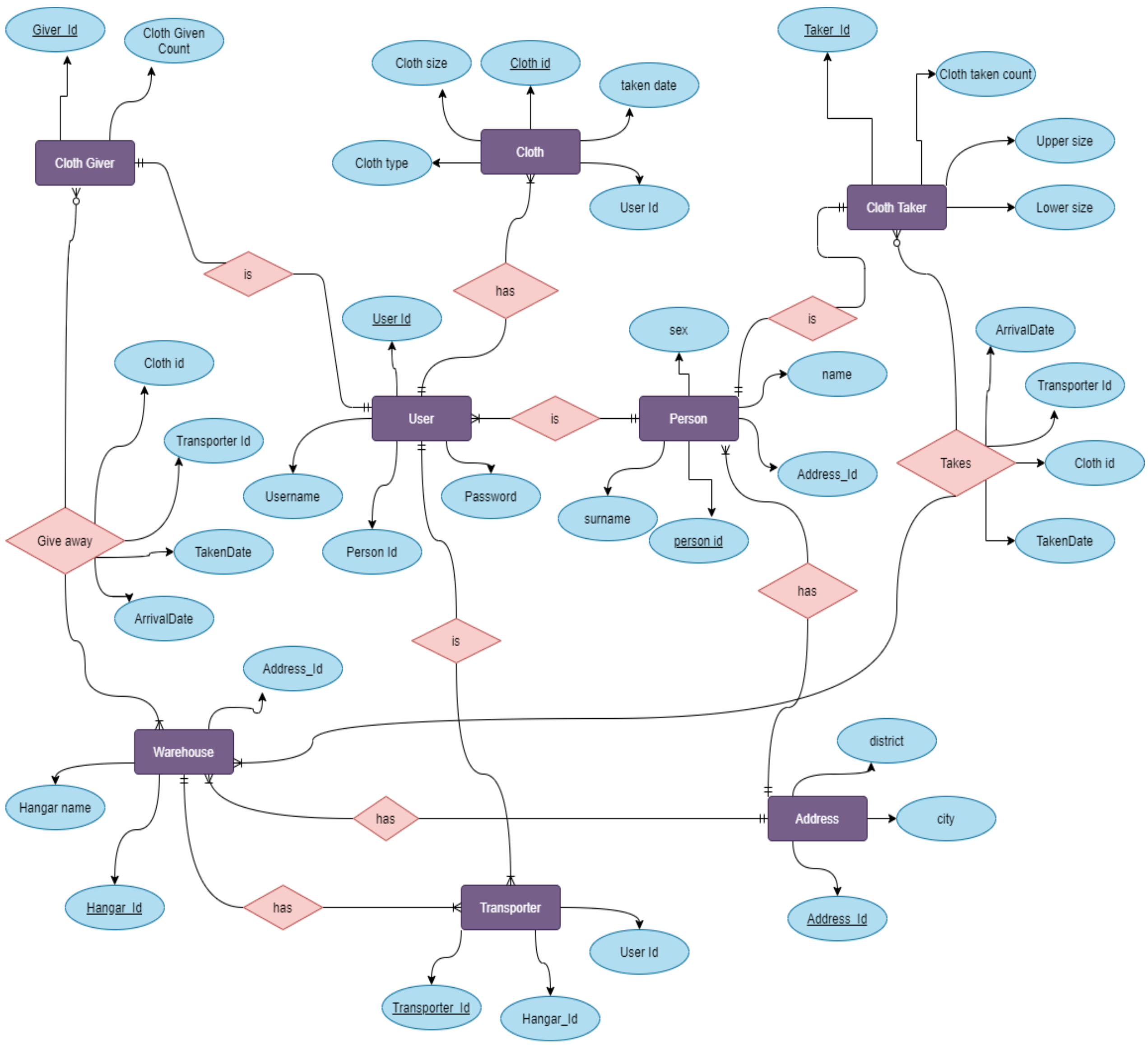


Figure2.: E-R Diagram

DESIGN PHILOSOPHY

ER Diagram:

- **User**
- **Person**
- **Clothe_giver**
- **Clothe_taker**
- **Cloth**
- **Hangar**
- **Transporter**
- **Adress**

In this part we will show the ER diagram to explain the whole system. By showing the ER diagram, sections that are not understood in the ER diagram will be understood.

User: gives its information to the **Clother_giver**, **Person**, **Cloth** and **Transporter**. Basically it keeps the general information of the user like: **username** and **password. User_id** will be generated automatically and with this way other entities takes user entity information's.

Person: Person class **name**,**surname sex** and **person_id(generated automatically).** Also there are another two subclass which are **Clothe_taker, Clothe_giver** sub classes, those are connected to the **User** and **Person** entities.

Clothe_giver: Clothe_giver is a sub-class of **User** entity. **Clothe_giver** keeps **user_id, cloth_given_count**. Finally it has a foreign key comes from User Entity.

Adress: Adress has connection between person and it keeps, **city** name and **district.**

Transporter: have connections between takes, hangar, user and giveaway. It keeps the **user_id** and **hangar_id** and **transporter_id.**

Cloth: is subclass of user. It **keeps cloth_id (generated automatically), user_id cloth_size, cloth_type, hangar_id.**

Hangar: keeps the information like **hangar_id(generated automatically),, address_id hangar_name.**

Clothe_Taker: Clothe_taker is a sub-class of **Person** entity. **Clothe_taker** keeps **person_id, lowerSize, upperSize and cloth_taken_count**. Finally it has a foreign key comes from Perosn Entity.

CARDINALITIES

MANY TO OPTIONAL MANY

Warehouse, Cloth Giver

Warehouse, Cloth Taker

ONE TO MANY

Person,User

User,Transporter

Warehouse, Transporter

Person, ClothTaker

User, Cloth Giver

Person, Cloth

Address,Person

Address,Hangar

USER PERMISSIONS

CLOTH GIVER

Add: User, Clothe type

Delete: User, Clothe type,

Update: User, Clothe type,

View: User, Clothe type, Warehouse

CLOTH TAKER

Add: User, Clothe type

Delete: User, Clothe type,

Update: User, Clothe type,

View: User, Clothe type, Warehouse

WAREHOUSE

Add: Hangar name

Delete: Hangar name

Update: Hangar name

View: Hangar name, Cloth giver, Cloth Taker

PHASE 2

NORMALIZATION

I will talk about the normalization process that we have done for a more consistent and stable operation of the database. In addition to stability, we envisioned preventing the parts that occupy space in the system due to duplicate data by developing a better design.

I will try to explain in detail what we are doing to achieve 1NF, the first step in normalization. As it is known in 1NF, one information should be stored in a cell in the database, if more than one data is stored, this situation does not comply with 1NF. We also had this problem at the design stage. Donations made and donations received by individuals can be considered as an example to reach 1NF. because a person may have made more than one donation or a person may own more than one garment at the same time, and these are data that cannot be kept in a single cell. Therefore, 1NF status has been tried to be provided by using takes and giveAway relations.

I will talk about another step, 2NF, in this section. As seen in our system, 1 person has more than one feature, but they are not kept in only 1 table. To give an example, the person in the system can also be a clothTaker person. Besides, this person can also be a user. and this user could be clothGiver or Transporter. In order to provide all these, we were able to talk about the same person with the help of a foreign key. We defined the primary key of one table as the foreign key for the primary key of the other table. To give an example, clothTaker is a person and this person can use his person information using his TakerId because the primary key that defines the ClothTaker actually has the same value as the primary key that has its person properties. The same relationship as seen in this system exists between ClothGiver and user. Transporter is also connected to the user with the same logic and the user is connected to the person with the key it contains. Thanks to the features I have explained here, the desired conditions are also met in 2NF.

Finally, I want to talk about what we did for the 3NF part. province example is related to address. As we know, city and district properties belong to the addressId, also known as the postal code. and this city and district information depends on the address id. As mentioned in the definition of 3NF, if 1 or more columns are connected to a column other than the primary key, a separate table can be opened for these columns and this column can be used as a primary key to define them. As a result, city and district data are linked to address id and address id is not primary key, so if we leave a single address id in person and store the values related to this id in another table, we achieve this goal. The second example is related to cloth. As we know, the data of a cloth does not change according to the owner of that cloth, it is completely dependent on the cloth. In this case, if this cloth information depends on the clothId rather than the personId, it would not be logical to keep this cloth information in the person, so we transferred the cloth information to another table and these data are stored here depending on the clothId.

Thanks to these steps, the system runs very stable, and whenever an update, delete, insert or update command is given, the system continues to work stably. At the same time, different information about a person is kept in different tables in the divided tables, and while this is done, the same data is not stored repeatedly in order not to fill the memory unnecessarily. Another thing is that we store all personal information in separate tables rather than in a single table, so if a person is not a transporter, we do not keep information about that he is not a transporter in vain. If we were to keep all the information on a table, whether a person is a transporter, whoever is a dresser or whoever is, obviously, no distinction could be made, so this kind of unnecessary information would be kept and memory would have been filled in vain. In summary, a quality database was created thanks to the normalization steps implemented.

E-R to Relational Mapping

- **Normal Entities**

Adress(Adress_id, District, City)

Person(Person_id, Name, Surname, Sex, Adress_id)

User(User_id,Person_id,username,password)

Hangar(Hangaar_id, HangarName,Adress_id)

Transporter(Transporter_id, User_id, Hangar_id)

Cloth(Cloth_id, User_id, ClothSize, ClothType, Hangar_id)

Clothgiver(ClothGiver_id, ClothGiven_id)

Clothtaker(ClothTaker_id, LowerSize, UpperSize, ClothTakenCount)

- **Relationships**

Takes(Takes_id, ArrivalDate, Transporter_id, Cloth_id, TakenDate, ClothTaker_id, Hangar_id)

Giveaway(Giveaway_id, Cloth_id, Transporter_id, TakenDate, ArrivalDate, ClothGiver_id, Hangar_id)

Functional Dependencies

Address(Adress_id, District, City)

Address_id → District

Address_id → City

Identification key: Address_id

Person(Person_id, Name, Surname, Sex, Address_id)

Person_id → Name

Person_id → Surname
Person_id → Sex
Person_id → Address_id
Person_id → District
Person_id → City
Identification key: Person_id

User(User_id,Person_id,username,password)
User_id → Person_id
User_id → username
User_id → password
User_id → Name
User_id → Surname
User_id → Sex
User_id → District
User_id → City
Identification key: User_id

Hangar(Hangaar_id, HangarName,Adress_id)
Hangar_id → HangarName
Hangar_id → address_id
Hangar_id → District
Hangar_id → City
Identification key: Hangar_id

Transporter(Transporter_id, User_id, Hangar_id)
Transporter_id → User_id
Transporter_id → Hangar_id
Transporter_id → Person_id
Transporter_id → username
Transporter_id → password
Transporter_id → Name
Transporter_id → Surname
Transporter_id → Sex
Transporter_id → Address_id
Transporter_id → District
Transporter_id → City
Identification key: Transporter_id

Cloth(Cloth_id, User_id, ClothSize, ClothType, Hangar_id)
Cloth_id → User_id
Cloth_id → ClothSize
Cloth_id → ClothType
Cloth_id → Hangar_id
Cloth_id → Person_id
Cloth_id → Name
Cloth_id → Surname
Cloth_id → Sex

Cloth_id → Address_id
Cloth_id → District
Cloth_id → City
Cloth_id → HangarName
Identification key: Cloth_id

Clothgiver(ClothGiver_id, ClothGiven_id)
ClothGiver_id -→ ClothGiven_id
ClothGiven_id → ClothGiver_id
Identification key: ClothGiven_id

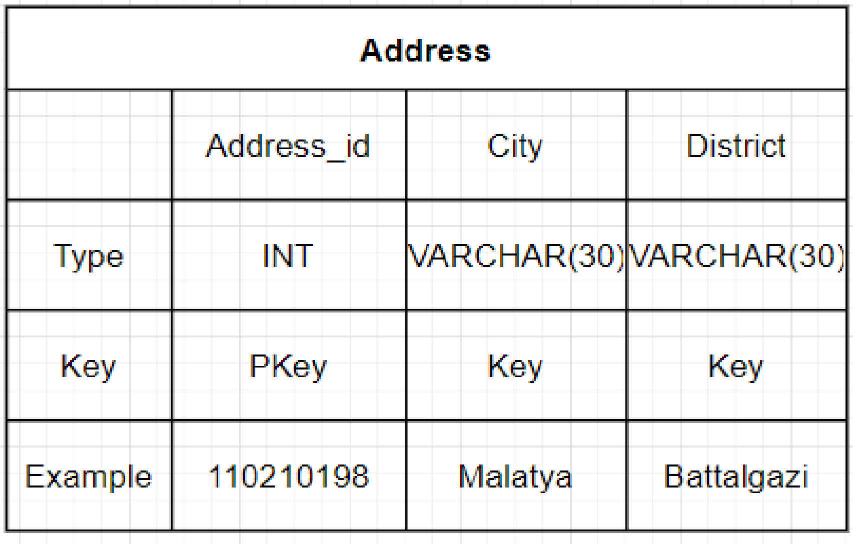
Clothtaker(ClothTaker_id, LowerSize, UpperSize, ClothTakenCount)
ClothTaker_id → LowerSize
ClothTaker_id → UpperSize
ClothTaker_id → ClothTakenCount
Identification key: ClothTaker_id

Takes(Takes_id, ArrivalDate, Transporter_id, Cloth_id, TakenDate, ClothTaker_id, Hangar_id)
Takes_id → ArrivalDate
Takes_id → Transporter_id
Takes_id → Cloth_id
Takes_id → TakenDate
Takes_id → ClothTaker_id
Takes_id → Hangar_id
Takes_id → User_id
Takes_id → Person_id

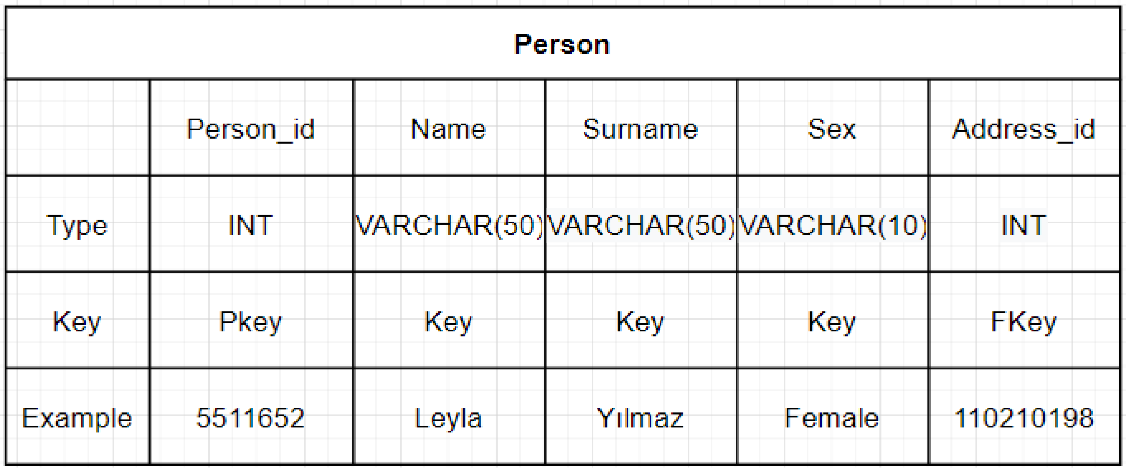
Takes_id → username
Takes_id → password
Takes_id → Name
Takes_id → Surname
Takes_id → Sex
Takes_id → Address_id
Takes_id → District
Takes_id → City
Takes_id → LowerSize
Takes_id → UpperSize
Takes_id → ClothTakenCount
Takes_id → HangarName
Identification key: Takes_id

Giveaway(Giveaway_id, Cloth_id, Transporter_id, TakenDate, ArrivalDate, ClothGiver_id, Hangar_id)
Giveaway_id → Cloth_id
Giveaway_id → Transporter_id
Giveaway_id → TakenDate
Giveaway_id → ArrivalDate
Giveaway_id → ClothGiver_id
Giveaway_id → Hangar_id
Giveaway_id → User_id
Giveaway_id → Person_id
Giveaway_id → username
Giveaway_id → password
Giveaway_id → Name
Giveaway_id → Surname
Giveaway_id → Sex
Giveaway_id → Address_id
Giveaway_id → District
Giveaway_id → City
Giveaway_id → ClothGiven_id
Giveaway_id → HangarName
Identification key: Giveaway_id

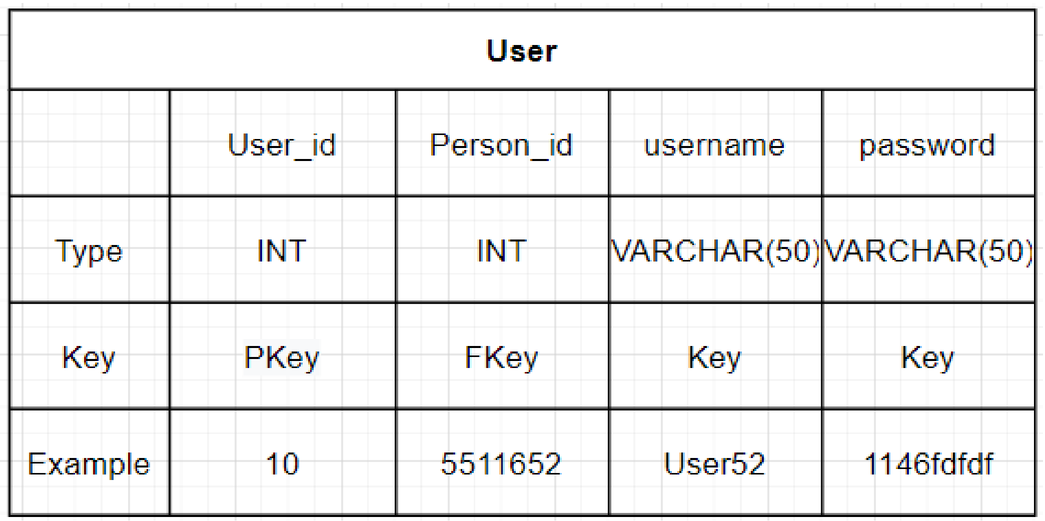
DATABASE SCHEMA



Address			
	Address_id	City	District
Type	INT	VARCHAR(30)	VARCHAR(30)
Key	PKey	Key	Key
Example	110210198	Malatya	Battalgazi

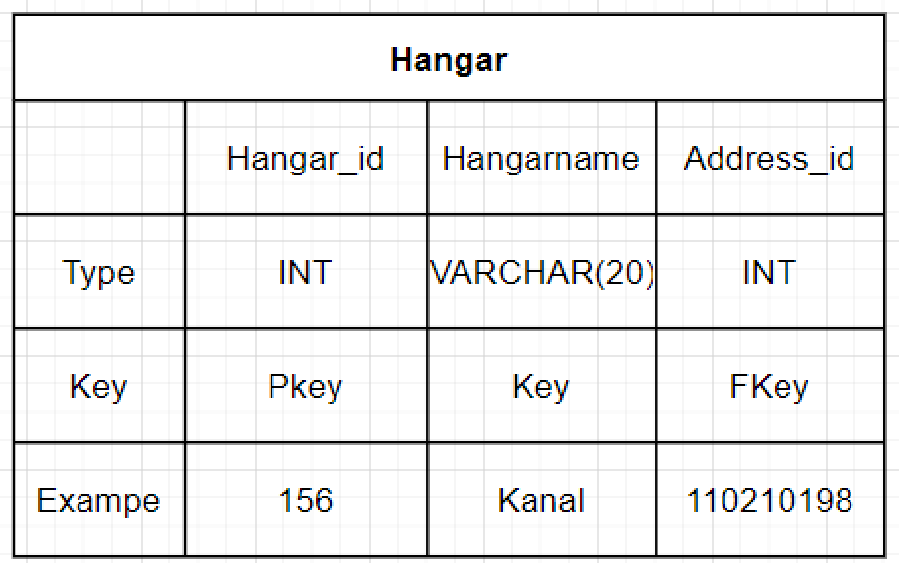


Person					
	Person_id	Name	Surname	Sex	Address_id
Type	INT	VARCHAR(50)	VARCHAR(50)	VARCHAR(10)	INT
Key	Pkey	Key	Key	Key	FKey
Example	5511652	Leyla	Yılmaz	Female	110210198



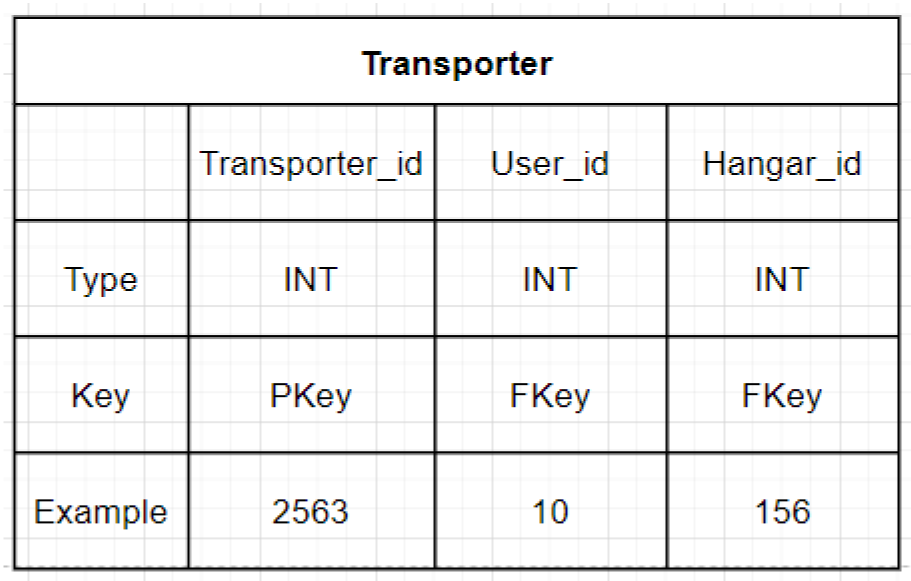
User

	User_id	Person_id	username	password
Type	INT	INT	VARCHAR(50)	VARCHAR(50)
Key	PKey	FKey	Key	Key
Example	10	5511652	User52	1146fdfdf



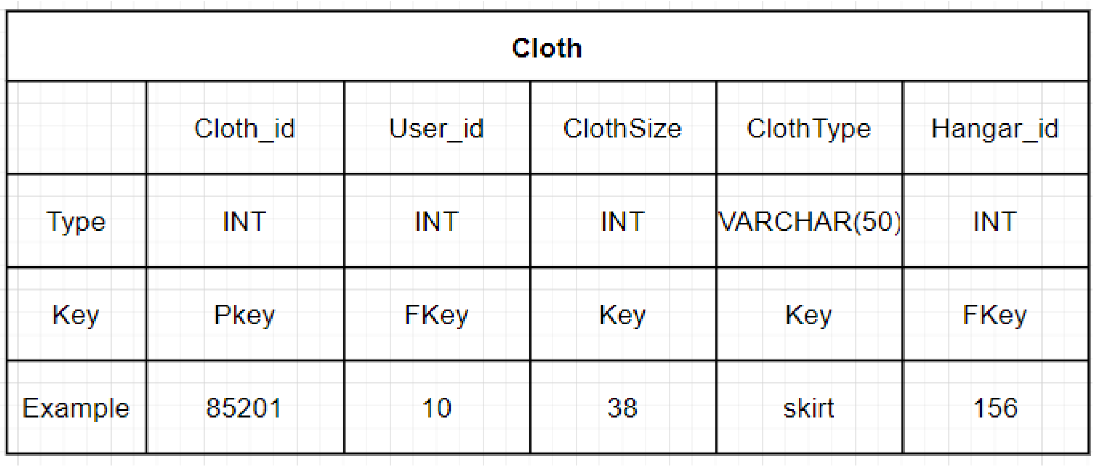
Hangar

	Hangar_id	Hangarname	Address_id
Type	INT	VARCHAR(20)	INT
Key	Pkey	Key	FKey
Exampe	156	Kanal	110210198



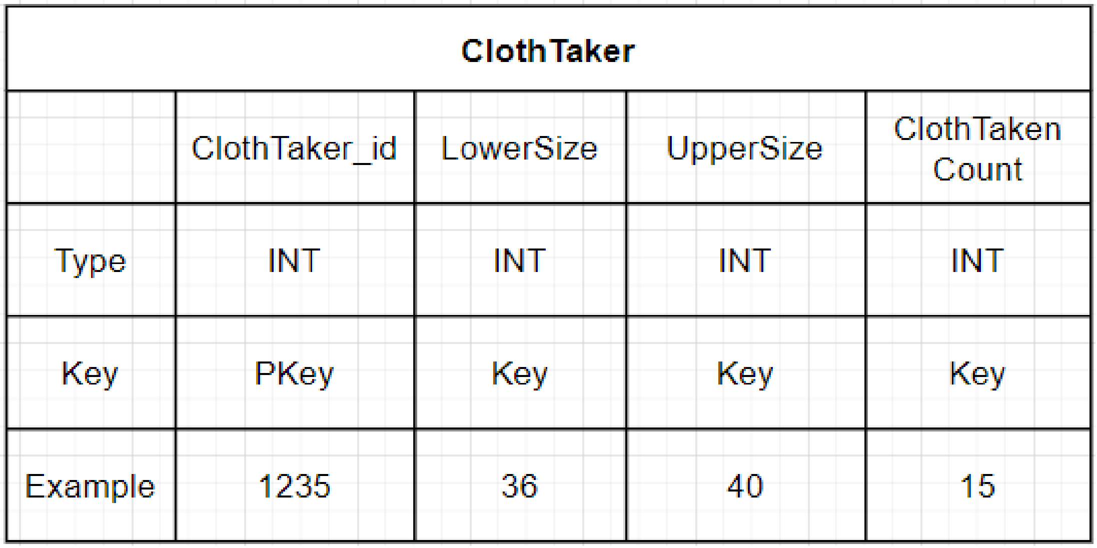
Transporter

	Transporter_id	User_id	Hangar_id
Type	INT	INT	INT
Key	PKey	FKey	FKey
Example	2563	10	156



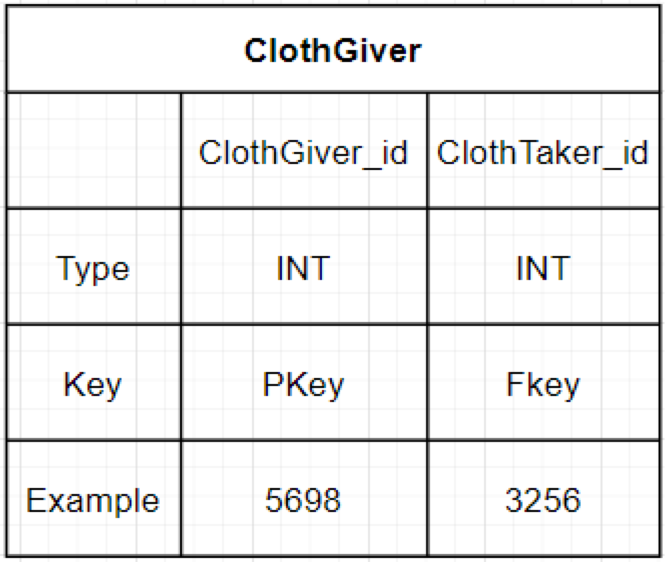
Cloth

	Cloth_id	User_id	ClothSize	ClothType	Hangar_id
Type	INT	INT	INT	VARCHAR(50)	INT
Key	Pkey	FKey	Key	Key	FKey
Example	85201	10	38	skirt	156



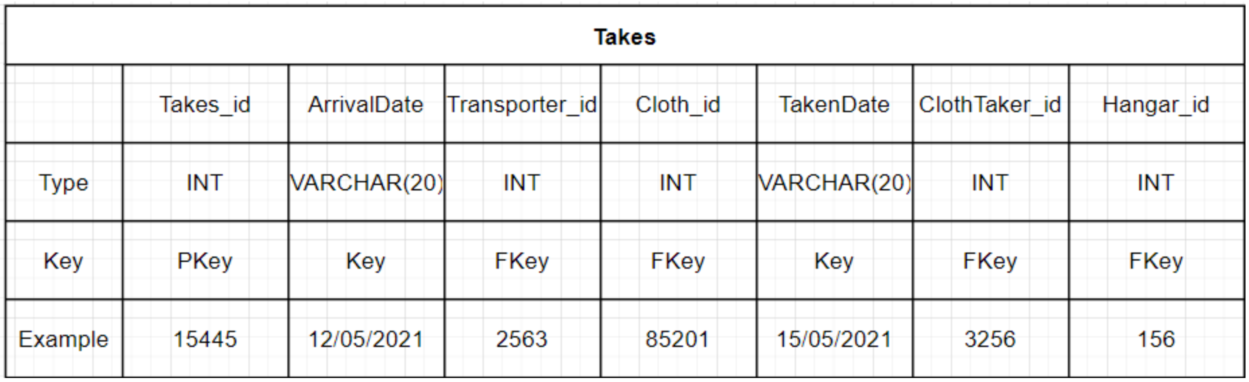
ClothTaker

	ClothTaker_id	LowerSize	UpperSize	ClothTaken Count
Type	INT	INT	INT	INT
Key	PKey	Key	Key	Key
Example	1235	36	40	15



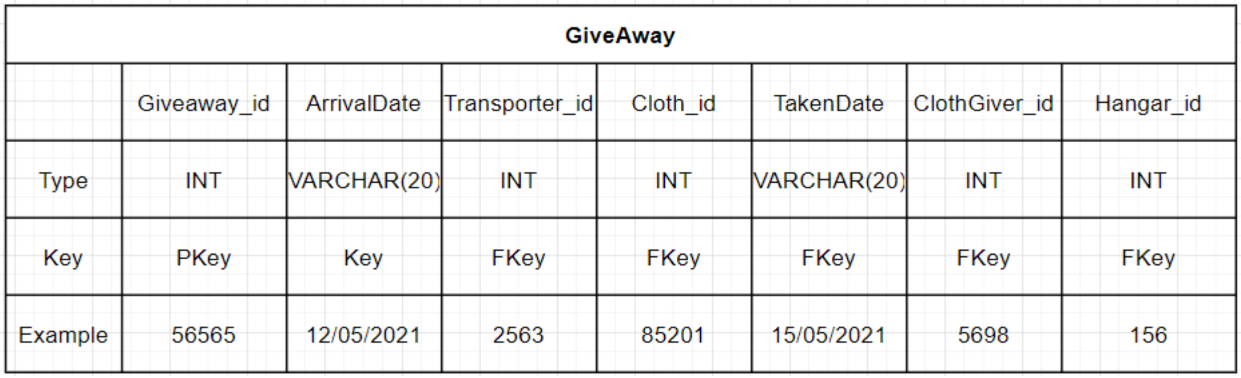
ClothGiver

	ClothGiver_id	ClothTaker_id
Type	INT	INT
Key	PKey	Fkey
Example	5698	3256



Takes

	Takes_id	ArrivalDate	Transporter_id	Cloth_id	TakenDate	ClothTaker_id	Hangar_id
Type	INT	VARCHAR(20)	INT	INT	VARCHAR(20)	INT	INT
Key	PKey	Key	FKey	FKey	Key	FKey	FKey
Example	15445	12/05/2021	2563	85201	15/05/2021	3256	156



GiveAway

	Giveaway_id	ArrivalDate	Transporter_id	Cloth_id	TakenDate	ClothGiver_id	Hangar_id
Type	INT	VARCHAR(20)	INT	INT	VARCHAR(20)	INT	INT
Key	PKey	Key	FKey	FKey	FKey	FKey	FKey
Example	56565	12/05/2021	2563	85201	15/05/2021	5698	156

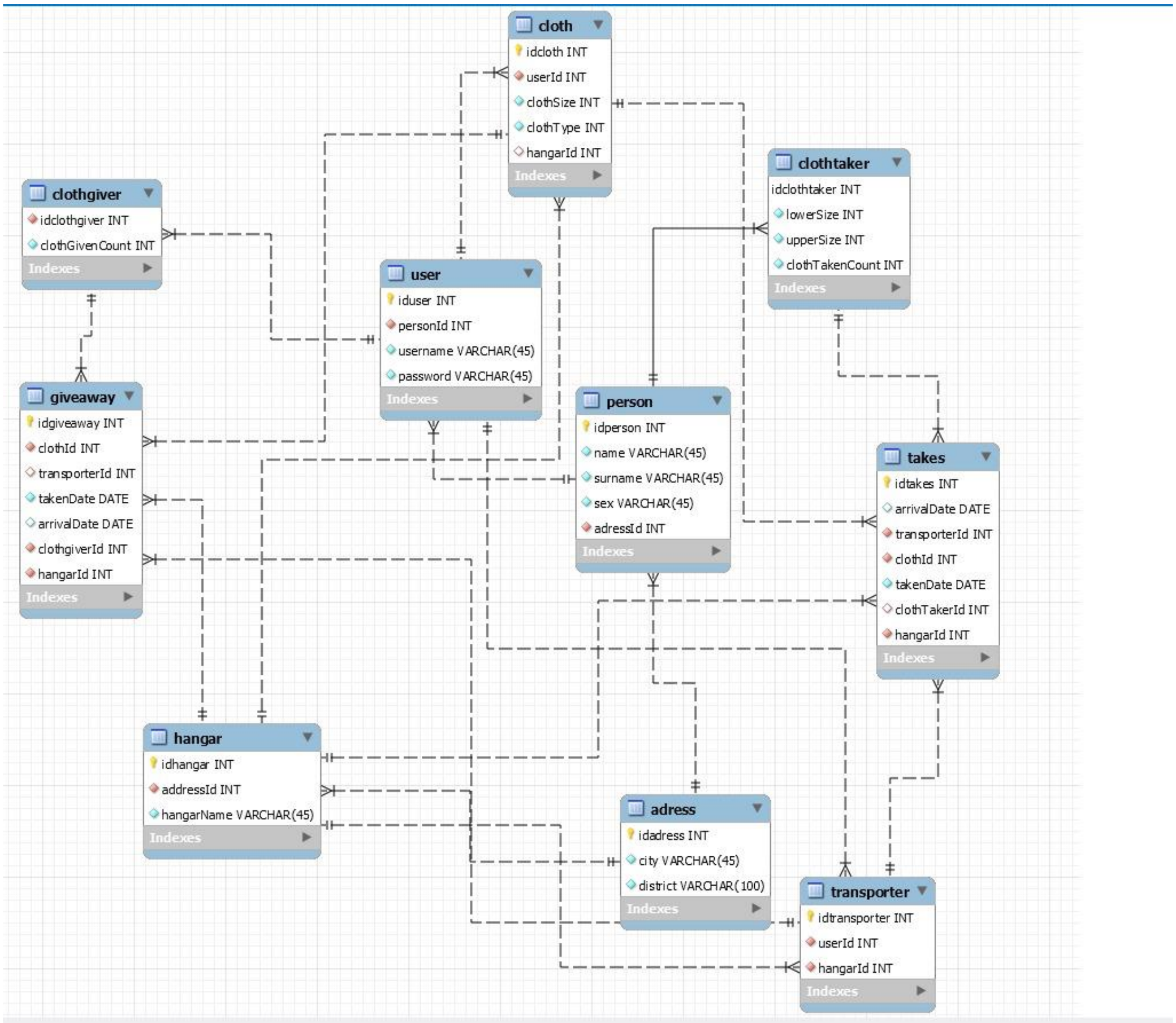


Figure3.: EER Database Schema

SCRIPTS

DDL CODES:

```
CREATE TABLE `adress` (
  `idadress` int NOT NULL,
  `city` varchar(45) NOT NULL,
  `district` varchar(100) NOT NULL,
  PRIMARY KEY (`idadress`)
)
;
```

```
CREATE TABLE `person` (
  `idperson` int NOT NULL,
  `name` varchar(45) NOT NULL,
  `surname` varchar(45) NOT NULL,
  `sex` varchar(45) NOT NULL,
  `adressId` int NOT NULL,
  PRIMARY KEY (`idperson`),
  KEY `person_FK1_idx` (`adressId`),
  CONSTRAINT `person_FK1` FOREIGN KEY (`adressId`) REFERENCES `adress` (`idadress`)
)
;

CREATE TABLE `hangar` (
  `idhangar` int NOT NULL,
  `addressId` int NOT NULL,
  `hangarName` varchar(45) NOT NULL,
  PRIMARY KEY (`idhangar`),
  KEY `hangar_FK1_idx` (`addressId`),
  CONSTRAINT `hangar_FK1` FOREIGN KEY (`addressId`) REFERENCES `adress`
(`idadress`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
;

CREATE TABLE `user` (
  `iduser` int NOT NULL,
  `personId` int NOT NULL,
  `username` varchar(45) NOT NULL,
  `password` varchar(45) NOT NULL,
  PRIMARY KEY (`iduser`),
  KEY `user_FK1_idx` (`personId`),
  CONSTRAINT `user_FK1` FOREIGN KEY (`personId`) REFERENCES `person` (`idperson`)
)
;

CREATE TABLE `cloth` (
  `idcloth` int NOT NULL,
  `userId` int NOT NULL,
  `clothSize` int NOT NULL,
  `clothType` int NOT NULL,
  `hangarId` int DEFAULT NULL,
  PRIMARY KEY (`idcloth`),
  KEY `cloth_FK1_idx` (`userId`),
  KEY `cloth_FK2_idx` (`hangarId`),
  CONSTRAINT `cloth_FK1` FOREIGN KEY (`userId`) REFERENCES `user` (`iduser`),
  CONSTRAINT `cloth_FK2` FOREIGN KEY (`hangarId`) REFERENCES `hangar` (`idhangar`)
)
```

```
;

CREATE TABLE `transporter` (
  `idtransporter` int NOT NULL,
  `userId` int NOT NULL,
  `hangarId` int NOT NULL,
  PRIMARY KEY (`idtransporter`),
  KEY `transporter_FK1_idx` (`userId`),
  KEY `transporter_FK2_idx` (`hangarId`),
  CONSTRAINT `transporter_FK1` FOREIGN KEY (`userId`) REFERENCES `user` (`iduser`),
  CONSTRAINT `transporter_FK2` FOREIGN KEY (`hangarId`) REFERENCES `hangar`
(`idhangar`)
)
;

CREATE TABLE `clothgiver` (
  `idclothgiver` int NOT NULL,
  `clothGivenCount` int NOT NULL,
  KEY `clothgiver_FK1_idx` (`idclothgiver`),
  CONSTRAINT `clothgiver_FK1` FOREIGN KEY (`idclothgiver`) REFERENCES `user`
(`iduser`)
)
;
CREATE TABLE `clothtaker` (
  `idclothtaker` int NOT NULL,
  `lowerSize` int NOT NULL,
  `upperSize` int NOT NULL,
  `clothTakenCount` int NOT NULL,
  PRIMARY KEY (`idclothtaker`),
  CONSTRAINT `clothTaker_FK1` FOREIGN KEY (`idclothtaker`) REFERENCES `person`
(`idperson`)
)
;
CREATE TABLE `giveaway` (
  `idgiveaway` int NOT NULL AUTO_INCREMENT,
  `clothId` int NOT NULL,
  `transporterId` int DEFAULT NULL,
  `takenDate` date NOT NULL,
  `arrivalDate` date DEFAULT NULL,
  `clothgiverId` int NOT NULL,
  `hangarId` int NOT NULL,
  PRIMARY KEY (`idgiveaway`),
  KEY `giveaway_FK1_idx` (`clothId`),
  KEY `giveaway_FK2_idx` (`transporterId`),
  KEY `giveaway_FK3_idx` (`clothgiverId`),
  KEY `giveaway_FK4_idx` (`hangarId`),
  CONSTRAINT `giveaway_FK1` FOREIGN KEY (`clothId`) REFERENCES `cloth` (`idcloth`),
  CONSTRAINT `giveaway_FK2` FOREIGN KEY (`transporterId`) REFERENCES `transporter`
(`idtransporter`),
  CONSTRAINT `giveaway_FK3` FOREIGN KEY (`clothgiverId`) REFERENCES `clothgiver`
(`idclothgiver`),
  CONSTRAINT `giveaway_FK4` FOREIGN KEY (`hangarId`) REFERENCES `hangar`
(`idhangar`)
)
;


CREATE TABLE `takes` (
  `idtakes` int NOT NULL AUTO_INCREMENT,
  `arrivalDate` date DEFAULT NULL,
  `transporterId` int NOT NULL,
  `clothId` int NOT NULL,
  `takenDate` date NOT NULL,
  `clothTakerId` int DEFAULT NULL,
  `hangarId` int NOT NULL,
  PRIMARY KEY (`idtakes`),
  KEY `takes_FK1_idx` (`clothId`),
  KEY `takes_FK2_idx` (`transporterId`),
  KEY `takes_FK3_idx` (`hangarId`),
  KEY `takes_FK4_idx` (`clothTakerId`),
  CONSTRAINT `takes_FK1` FOREIGN KEY (`clothId`) REFERENCES `cloth` (`idcloth`),
  CONSTRAINT `takes_FK2` FOREIGN KEY (`transporterId`) REFERENCES `transporter`
(`idtransporter`),
  CONSTRAINT `takes_FK3` FOREIGN KEY (`hangarId`) REFERENCES `hangar` (`idhangar`),
  CONSTRAINT `takes_FK4` FOREIGN KEY (`clothTakerId`) REFERENCES `clothtaker`
(`idclothtaker`)
)
;
```