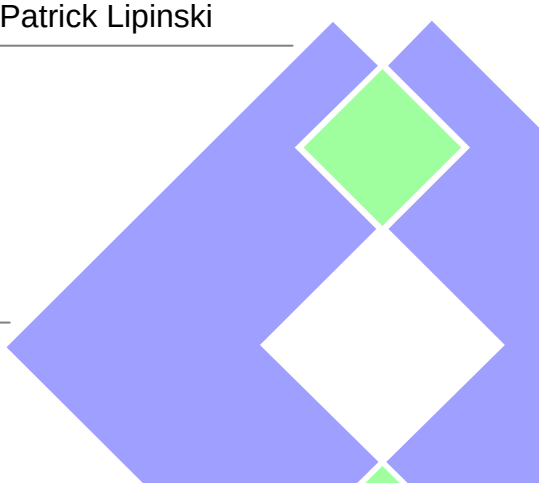
Datenbanken

Übungsaufgabe 2

Prof. Dr. Hohmann

Oliver Beckenhaub
Stefan Eyerich
Sebastian Hemel
Patrick Lipinski



Inhaltsverzeichnis

Allgemeine Aufgabenstellung:

Entwerfen Sie zu folgenden Problemstellungen ein geeignetes semantisches (Entity-Relationship-Diagramm) und logisches, relationales Datenbankkonzept. Soweit notwendig machen Sie Annahmen um die Aufgabenstellung zu lösen. Implementieren Sie das logische Datenbankkonzept und schreiben Sie SQL-Befehle für die häufigsten Auswertungen. Die Lösungen für die Aufgaben müssen Sie schriftlich ausarbeiten und mit Name und Matrikelnummer versehen abgeben. Verwenden Sie für die grafische Lösung Visio oder Oracle-Schema-Bilder.

1. Autohausinformationssystem

1.1 Aufgabenstellung

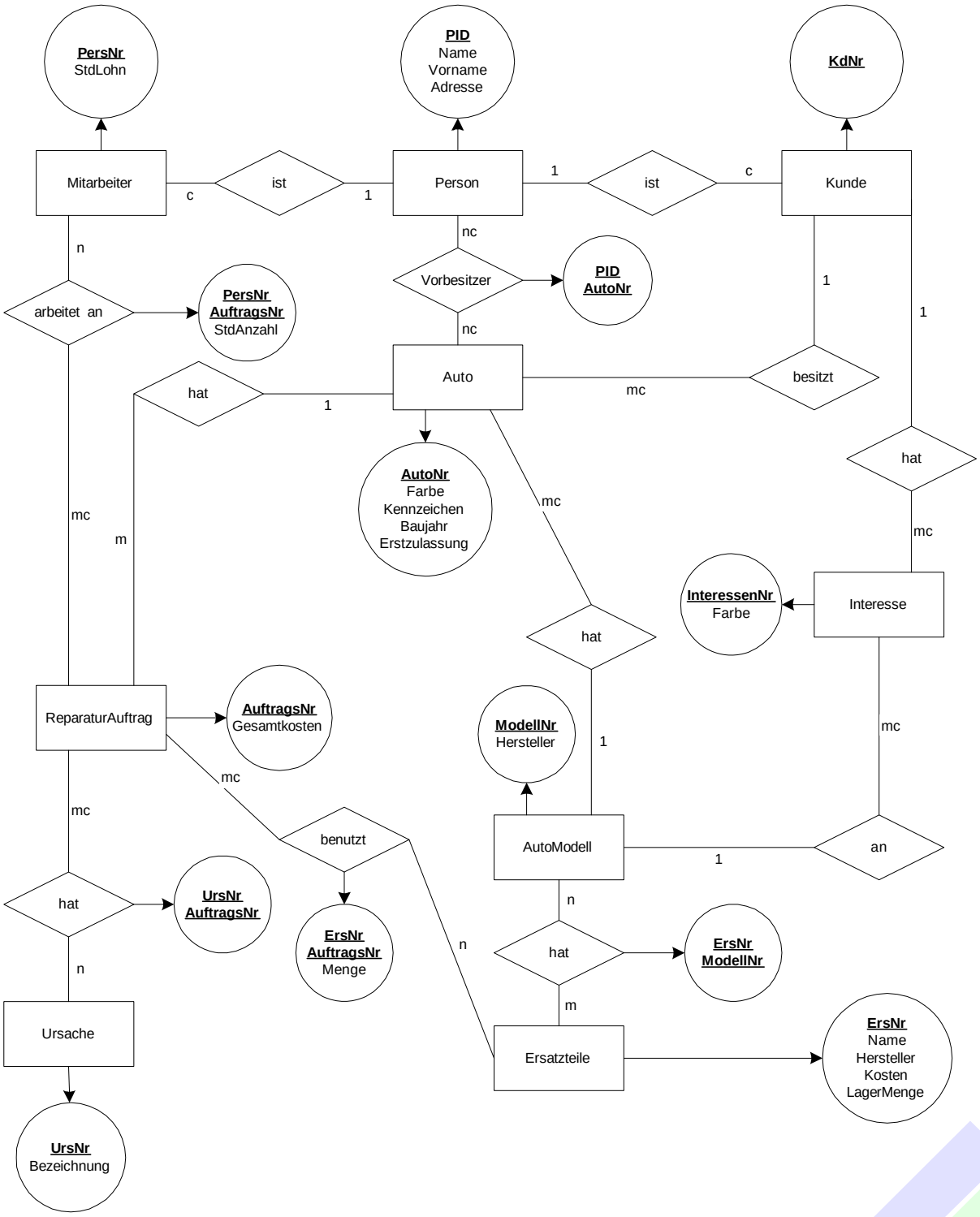
Die KFZ-Werkstatt „OPALO“ hat erkannt, dass es für den erfolgreichen Kundenkontakt notwendig ist, eine zentrale Datenbank mit Informationen über das Kundenfahrzeug, den Kunden, Reparaturen und Kauf-Interessen zu führen. Als wichtige Informationen für ein solches CRM-System sollen geführt werden:

- KFZ-Typ und weitere relevante Daten (z.B. Baujahr, Erstzulassung, Zulassungskennzeichen, Finanzierungsform, Farbe, Vorbesitzer) über die Kundenfahrzeuge; Selbstverständlich kann eine Kunde mehrere Fahrzeuge besitzen.
- Informationen über alle Reparaturen mit Termin, Rechnungsbetrag, verwendeten Ersatzteile und Klassifizierung von Reparatururs achen.
- Speicherung von Interessen und Verkaufschancen für Neu- und Gebrauchtfahrzeuge. Vorlieben des Kunden für bestimmte Wagentypen.

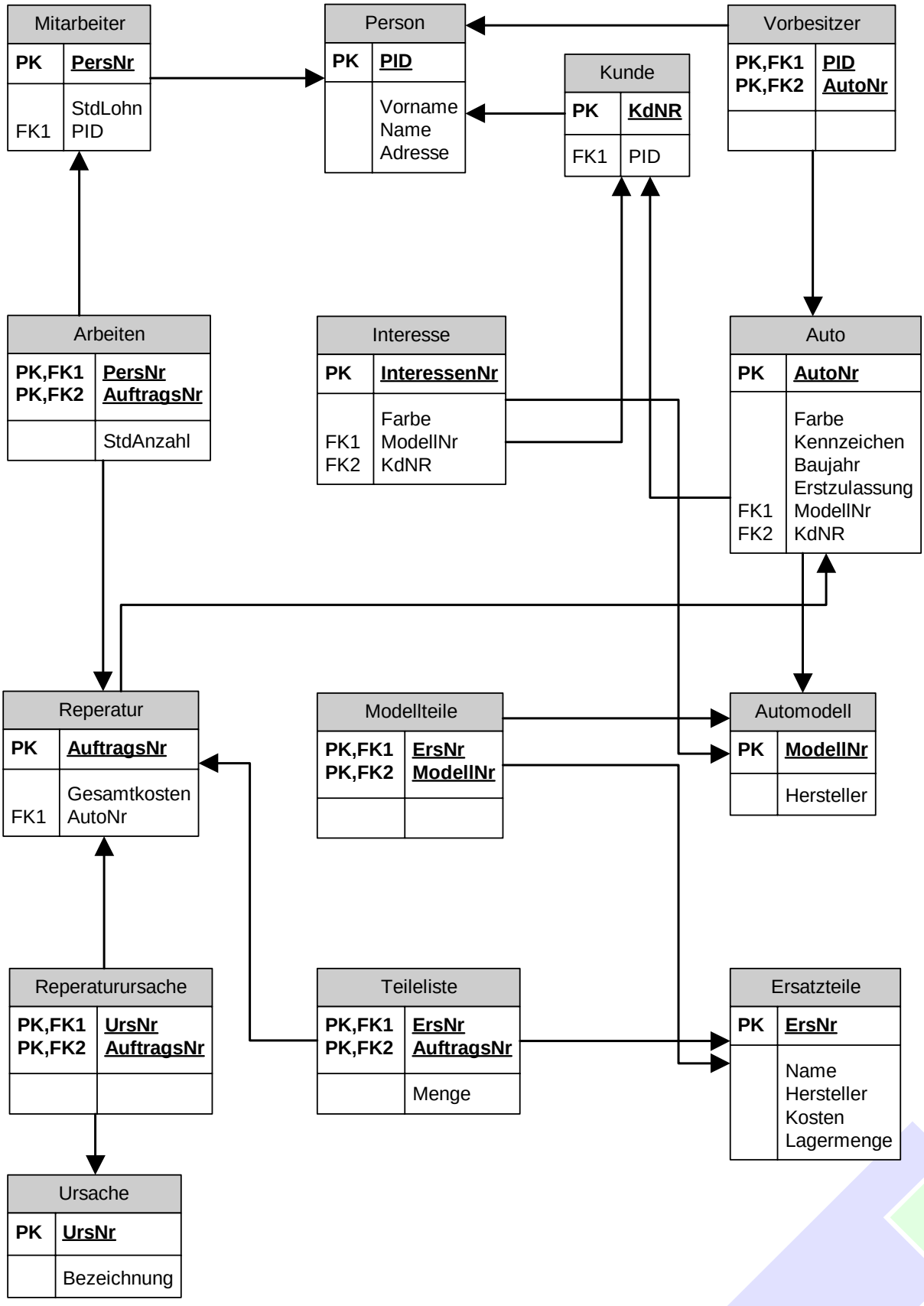
Mit Hilfe der so gespeicherten Daten sollen die folgenden, häufig vorkommenden, unterstützenden Suchfragen, erfüllt werden:

- Auskunft über die Anzahl der Werkstattbesuche eines bestimmten Kunden und Summierung der Reparaturkosten.
- Reparaturhäufigkeit der einzelnen Automarken und –modelle
- Erstellen einer Mailingliste für Kunden, die Fahrzeuge älter als 3 Jahre besitzen. Hierbei sollen die Interessenprofile der Kunden einbezogen werden.
- Abfrage der mittleren Reparaturkosten je Automodell.
- Liste mit den Summen an Ersatzteilen pro Fahrzeugtyp.

1.2 Entity-Relationship-Diagramm



1.3 Logisches Datenbankkonzept



1.4 SQL-Befehle (MS-SQL – Access)

1.4.1 Auskunft über die Anzahl der Werkstattbesuche eines bestimmten Kunden und Summierung der Reparaturkosten.

```
SELECT
        Kunde.KdNr,
        Count(Reparatur.AuftragsNr) AS AnzahlvonAuftragsNr,
        Sum(Reparatur.Gesamtkosten) AS SummevonGesamtkosten
FROM
        (Kunde INNER JOIN Auto ON Kunde.KdNr = Auto.KdNr)
        INNER JOIN Reparatur ON Auto.AutoNr = Reparatur.AutoNr
GROUP BY
        Kunde.KdNr;
```

	KdNr	AnzahlvonAuftragsNr	SummevonGesamtkosten
▶	2	3	80
	3	1	100

1.4.2 Reparaturhäufigkeit der einzelnen Automarken und –modelle

```
SELECT
        Automodell.ModellNr,
        Count(Reparatur.AuftragsNr) AS AnzahlvonAuftragsNr
FROM
        (Automodell INNER JOIN Auto ON Automodell.ModellNr = Auto.ModellNr)
        INNER JOIN Reparatur ON Auto.AutoNr = Reparatur.AutoNr
GROUP BY
        Automodell.ModellNr;
```

	ModellNr	AnzahlvonAuftragsNr
▶	1	3
	3	1

1.4.3 Erstellen einer Mailingliste für Kunden, die Fahrzeuge älter als 3 Jahre besitzen. Hierbei sollen die Interessenprofile der Kunden einbezogen werden.

```
SELECT
        Kunde.KdNr,
        Auto.Baujahr,
        Interesse.ModellNr,
        Interesse.Farbe
FROM
        (Kunde INNER JOIN Auto ON Kunde.KdNr = Auto.KdNr)
        Left JOIN Interesse ON Kunde.KdNr = Interesse.KdNr
WHERE
        (((Auto.Baujahr)<(Val(Year(Now()))-3)));
```

	KdNr	Baujahr	ModellNr	Farbe
▶	2	1972		
	3	1998	3	schwarz
	2	1999		

1.4.4 Abfrage der mittleren Reparaturkosten je Automodell.

```
SELECT
        Automodell.ModellNr,
        AVG(Reparatur.Gesamtkosten) AS MittlereReparaturkosten
FROM
        (Automodell INNER JOIN Auto ON Automodell.ModellNr = Auto.ModellNr)
        INNER JOIN Reparatur ON Auto.AutoNr = Reparatur.AutoNr
GROUP BY
        Automodell.ModellNr;
```

	ModellNr	MittlereReparaturkosten
▶	1	53,3333333333333
	3	20

1.4.5 Liste mit den Summen an Ersatzteilen pro Fahrzeugtyp.

```
SELECT
        Automodell.ModellNr,
        Sum(Ersatzteile.Lagermenge) AS SummevonLagermenge
FROM
        Ersatzteile INNER JOIN (Automodell INNER JOIN Modellteile ON
        Automodell.ModellNr = Modellteile.ModellNr) ON
        Ersatzteile.ErsNr = Modellteile.ErsNr
GROUP BY
        Automodell.ModellNr;
```

	ModellNr	SummevonLagermenge
▶	1	49
	2	15

2. Hochschulverwaltungssystem

2.1 Aufgabenstellung

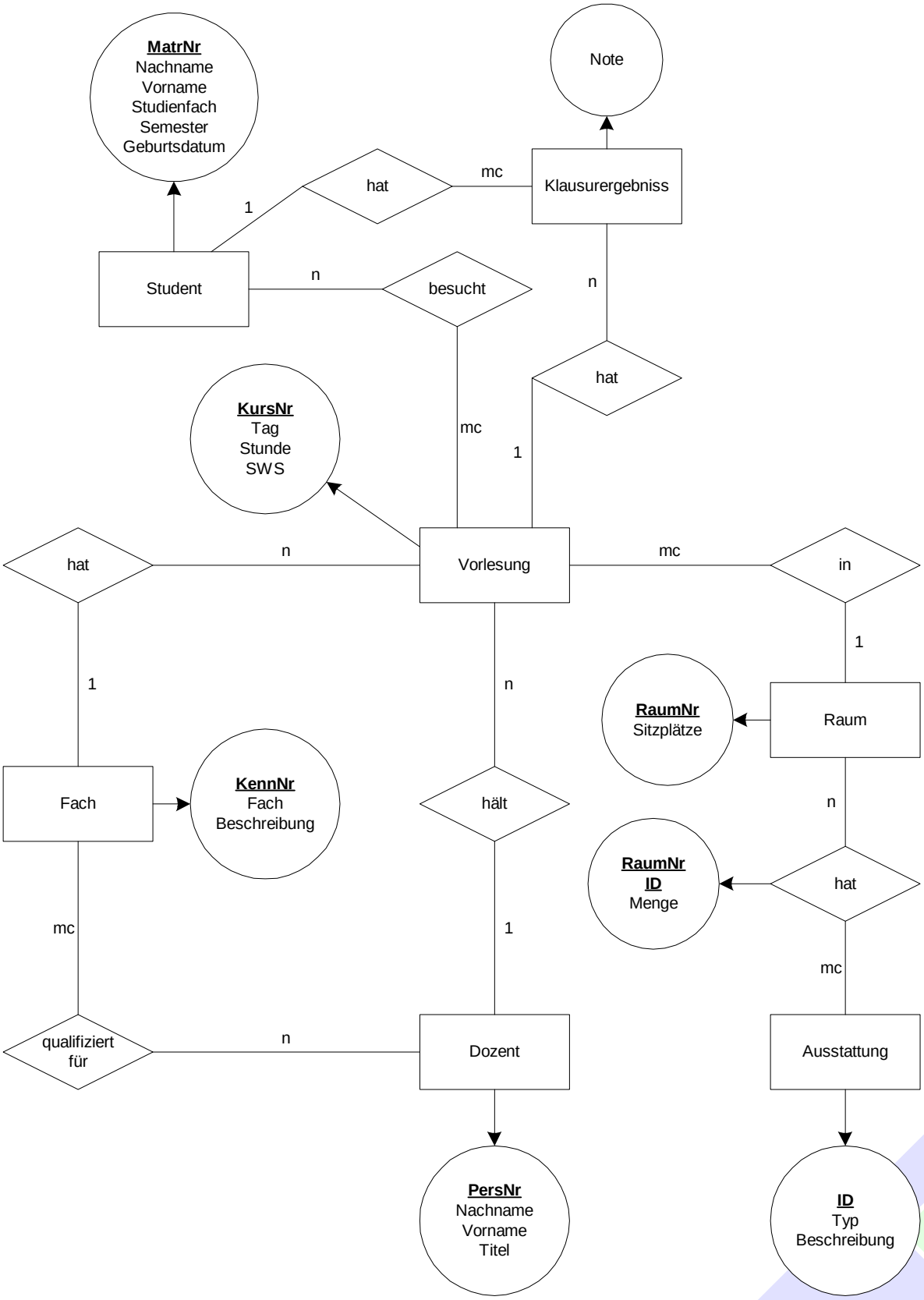
Zur besseren und schnelleren Verwaltung der Veranstaltungen, der Belegungen von Veranstaltungen und Prüfungsergebnisse der Studenten soll eine Datenbank eingesetzt werden. Hierzu ist es notwendig, die Studierenden der Hochschule (Name, Vorname, Matrikel-Nr., Studienfach, Semesteranzahl und Geburtsdatum), die Dozenten (Name, Qualifikation usw.), die Räume (Raumnummer, Ausstattung usw.) und Prüfungen zu speichern. Wesentliche Anforderungen sind:

- Zu Beginn des Semesters belegt ein Student eine Veranstaltung und meldet sich so automatisch auch für die Prüfung an.
- Die Prüfungsergebnisse werden nach erfolgter Prüfung gespeichert, wobei zu beachten ist, dass eine Prüfung maximal 3-mal geschrieben werden kann.
- Eine Veranstaltung wird pro Semester von einem oder mehreren Dozenten angeboten.
- Die Zuteilung der Räume soll zukünftig nach der Anmeldefrist automatisch erfolgen können, wobei zu berücksichtigen ist, dass die entsprechender Raumgröße und technische Ausstattung (Beamer, Overheadprojektor usw.) in den Räumen zur Verfügung stehen.

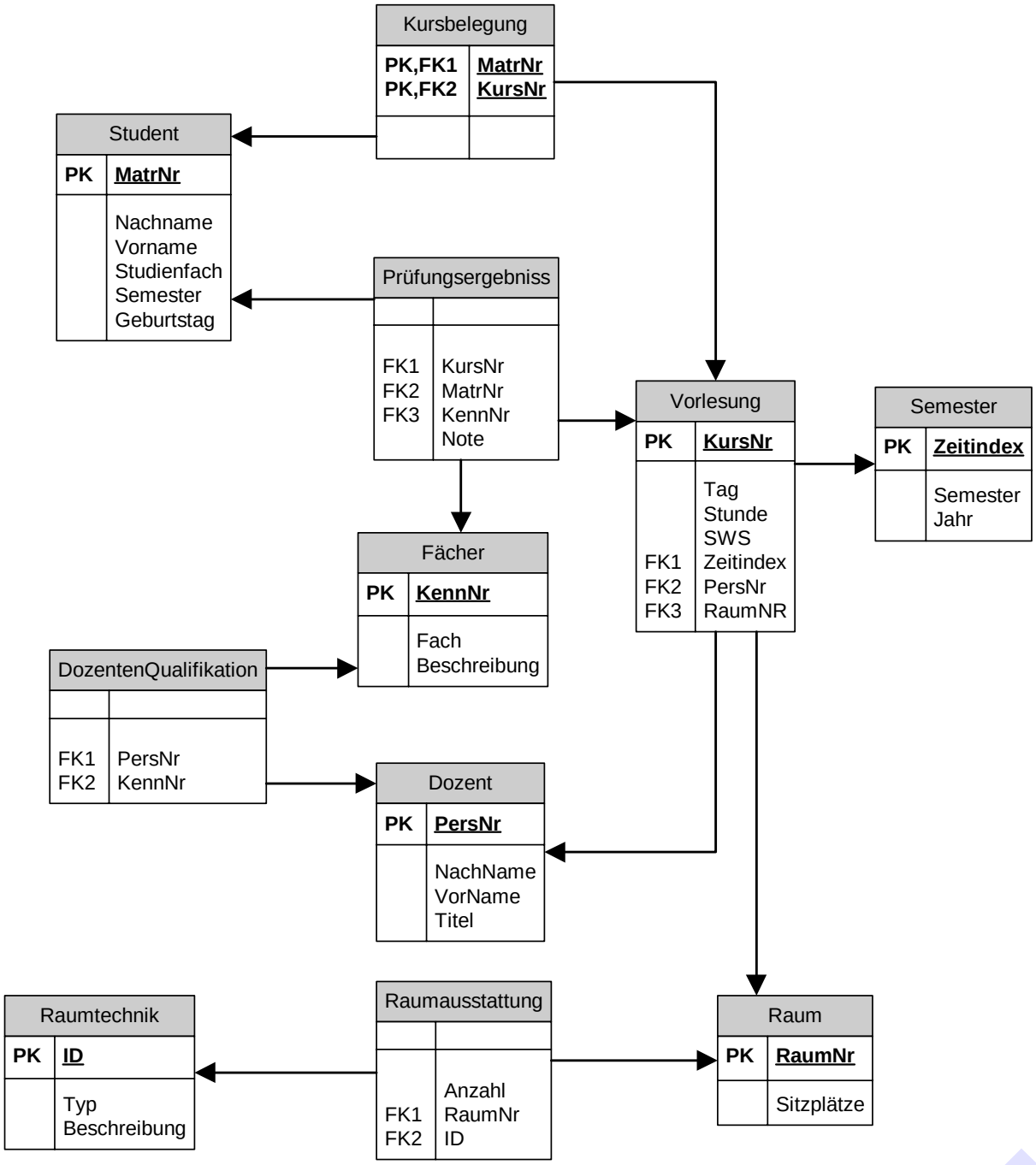
Häufige Auswertungen sind z.B.:

- Auskunft über die Studierenden, die eine Veranstaltung belegen.
- Durchschnittliche Studentenzahl pro Veranstaltung.
- Wie viele Lehrveranstaltungen hat ein Studierender belegt und wie viele Semesterwochenstunden kommen pro Semester zusammen.
- Wie häufig hat ein Studierender eine Klausur pro Fach geschrieben?
- Welcher Dozent gibt welche Veranstaltung?
- Wie sind die Räume belegt?

2.2 Entity-Relationship-Diagramm



2.3 Logisches Datenbankkonzept



2.4 SQL-Befehle (MS-SQL – Access)

2.4.1 Auskunft über die Studierenden, die eine Veranstaltung belegen.

```
SELECT
        Vorlesungen.KursNr,
        Kursbelegung.Student
FROM
        Vorlesungen INNER JOIN Kursbelegung ON
        Vorlesungen.KursNr = Kursbelegung.Kurs;
```

	MatrNr	Fach	KursNr
▶	2	DB	1
	3	DB	1
	4	DB	1
	5	DB	1
	3	GD	20
	1	GD	20
	3	DB	3
	8	DB	3
*			

2.4.2 Durchschnittliche Studentenzahl pro Veranstaltung.

```
SELECT
        Avg(AnzahlvonStudent) AS StudentproVeranstaltung
FROM
        [SELECT
                Count(Kursbelegung.Student) AS AnzahlvonStudent
        FROM
                Kursbelegung
        GROUP BY
                Kursbelegung.Kurs
        ]. AS [%$##@_Alias];
```

	StudentproVeranstaltung
▶	2,6666666666667

2.4.3 Wie viele Lehrveranstaltungen hat ein Studierender belegt und wie viele Semesterwochenstunden kommen pro Semester zusammen.

```
SELECT
        Student.MatrNr,
        Sum(Vorlesungen.SWS) AS SummevonSWS,
        Count(Vorlesungen.KursNr) AS AnzVorlesungen
FROM
        Vorlesungen INNER JOIN (Student INNER JOIN Kursbelegung ON
        Student.MatrNr = Kursbelegung.Student) ON Vorlesungen.KursNr =
        Kursbelegung.Kurs
GROUP BY
        Student.MatrNr, Vorlesungen.ZeitIndex
HAVING
        (((Vorlesungen.ZeitIndex)=0));
```

	MatrNr	SummevonSWS	AnzVorlesunge
▶	1	3	1
	2	2	1
	3	6	3
	4	2	1
	5	2	1
	8	1	1

2.4.4. Wie häufig hat ein Studierender eine Klausur pro Fach geschrieben?

```
SELECT
        Student.Nachname,
        Student.Vorname,
        Prüfungsergebnisse.MatrNr,
        Fächer.Fach,
        Count(Fächer.Fach) AS AnzahlvonFach
FROM
        Student INNER JOIN (Fächer INNER JOIN Prüfungsergebnisse ON
        Fächer.KennNr = Prüfungsergebnisse.Fach) ON Student.MatrNr =
        Prüfungsergebnisse.MatrNr
GROUP BY
        Student.Nachname,
        Student.Vorname,
        Prüfungsergebnisse.MatrNr,
        Fächer.Fach;
```

	Nachname	Vorname	MatrNr	Fach	AnzahlvonFach
▶	c	a	9	SE	1
	c	c	3	DB	1
	c	c	3	SE	2
	e	e	5	DB	1
	h	h	8	GD	1

2.4.5 Welcher Dozent gibt welche Veranstaltung?

```
SELECT
        Dozent.VorName,
        Dozent.NachName,
        Vorlesungen.KursNr,
        Fächer.Fach
FROM
        Fächer INNER JOIN (Dozent INNER JOIN Vorlesungen ON Dozent.PersNr =
        Vorlesungen.Dozent) ON Fächer.KennNr = Vorlesungen.Fach
WHERE
        (((Dozent.PersNr)=1));
```

	VorName	NachName	KursNr	Fach
▶	DA	DA	20	GD
	DA	DA	3	DB
	DA	DA	2	SE
*				

2.4.6 Wie sind die Räume belegt?

```
SELECT
        Raum.RaumNr,
        Vorlesungen.KursNr,
        Fächer.Fach
FROM
        Fächer INNER JOIN (Raum INNER JOIN Vorlesungen ON Raum.RaumNr =
        Vorlesungen.RaumNr) ON Fächer.KennNr = Vorlesungen.Fach
WHERE
        (((Raum.RaumNr)=11));
```

	RaumNr	KursNr	Fach
▶	11	1	DB
	11	20	GD
*			

3. Flugverbindungen

3.1 Aufgabenstellung

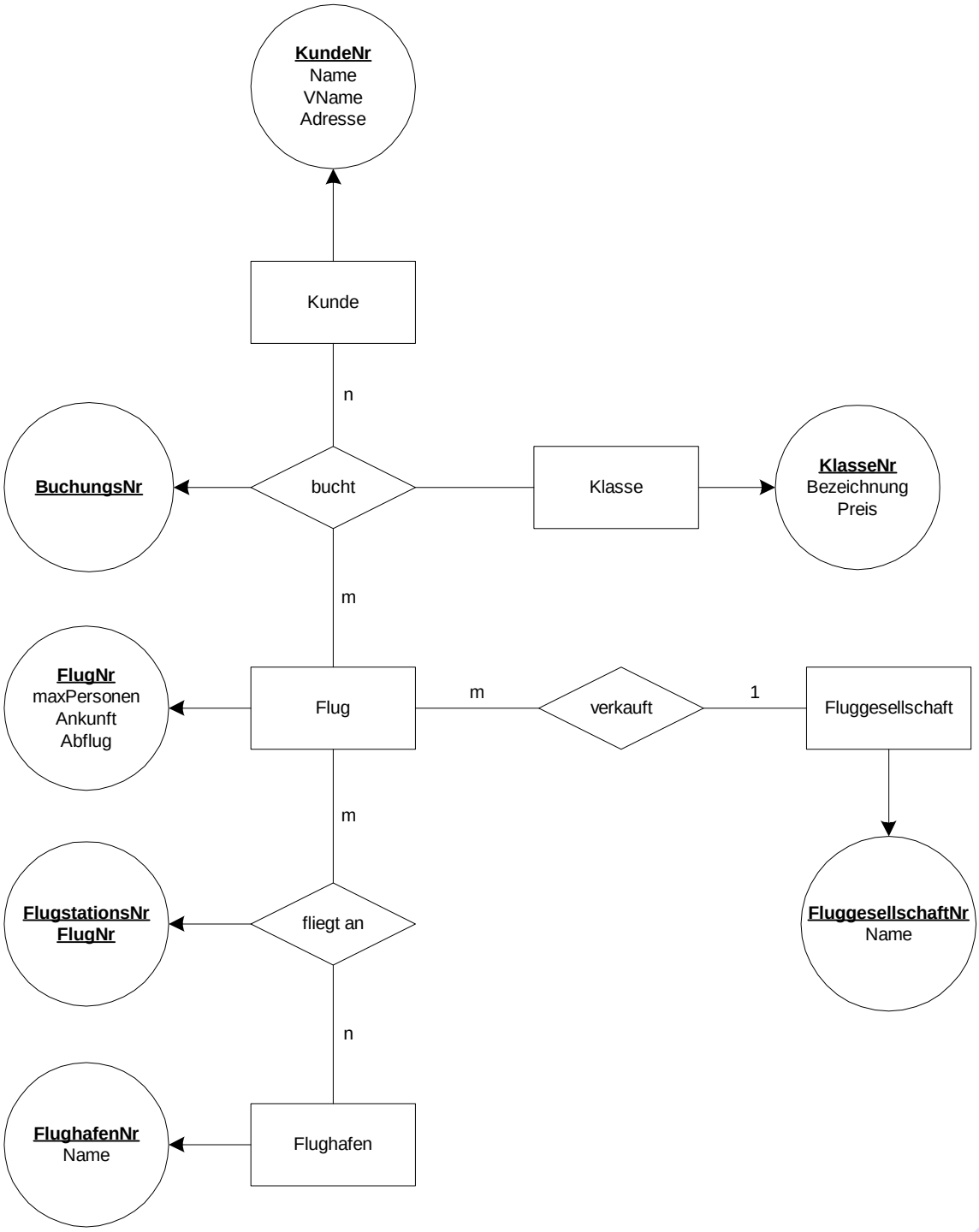
Entwerfen Sie ein Fluginformationssystem mit dem es möglich ist, Flüge, Fluggesellschaften, Buchungen und Flughäfen in der Datenbank zu speichern. Im Einzelnen soll folgendes möglich sein:

- Flüge müssen mit Zwischenstopps auf unterschiedlichen Flughäfen gespeichert werden können.
- Flüge müssen Details zur Abflugzeit, Ankunftszeit, Preise usw. enthalten.
- Fluggesellschaften verkaufen die Flüge an Kunden (Buchungen). Bei den Buchungen können bis zu drei Kategorien unterschieden werden.

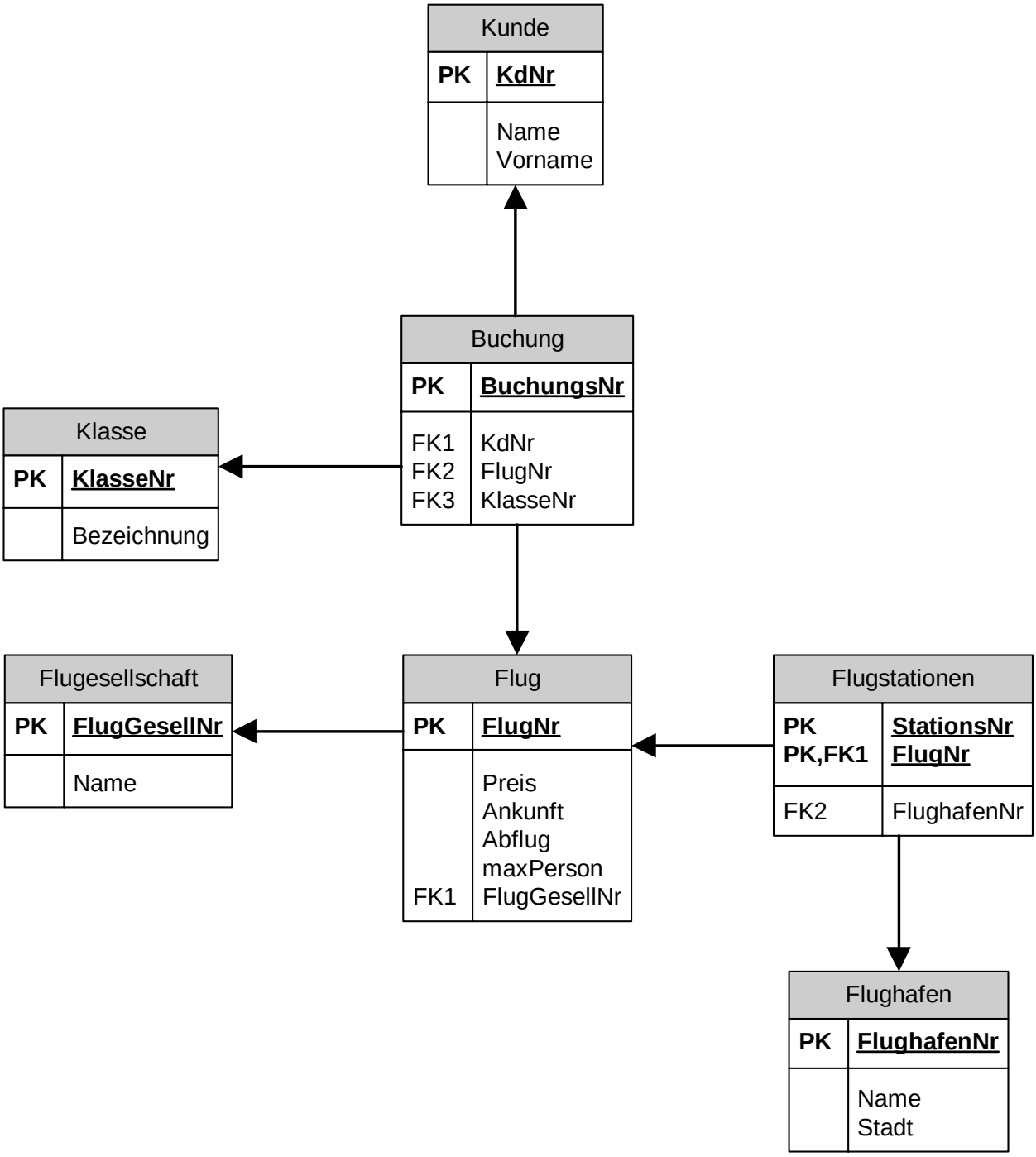
Häufige Auswertungen sind z.B.:

- Auskunft über die Auslastung der Flüge.
- Anzahl der Zwischenstopps von Flügen.
- Durchschnittliche Anzahl an Fluggästen pro Flug.
- Umsatz pro Flug.
- Welche Flughäfen werden von welchen Fluggesellschaften angeflogen?

3.2 Entity-Relationship-Diagramm



3.3 Logisches Datenbankkonzept



3.4 SQL-Befehle (MS-SQL – Access)

3.4.1 Auskunft über die Auslastung der Flüge.

```
SELECT
	tbl_Flug.FlugNr, ([a].[AnzFlug]/[maxPersonen])*100 AS [Auslastung in %]
FROM
	tbl_Flug INNER JOIN
		[
		SELECT
			tbl_Buchung.FlugNr,
			Count(tbl_Buchung.FlugNr) AS AnzFlug
		FROM
			tbl_Buchung
		GROUP BY
			tbl_Buchung.FlugNr
		]. AS a ON tbl_Flug.FlugNr = a.FlugNr;
```

	FlugNr	Auslastung in %
▶	1	10
	2	40
	3	10

3.4.2 Anzahl der Zwischenstopps von Flügen

```
SELECT
	tbl_Flugstationen.FlugNr,
	Count(tbl_Flugstationen.FlugNr) AS AnzahlvonFlugNr
FROM
	tbl_Flugstationen
GROUP BY
	tbl_Flugstationen.FlugNr;
```

	FlugNr	AnzahlvonFlugNr
▶	1	2
	2	4
	3	2
	4	1

3.4.3 Durchschnittliche Anzahl an Fluggästen pro Flug

```
SELECT
        Avg(a.AnzFlug) AS MittelwertvonAnzFlug
FROM
        tbl_Flug INNER JOIN [
                SELECT
                        tbl_Buchung.FlugNr,
                        Count(tbl_Buchung.FlugNr) AS AnzFlug
                FROM
                        tbl_Buchung
                GROUP BY
                        tbl_Buchung.FlugNr
                ]. AS a ON tbl_Flug.FlugNr = a.FlugNr;
```

	MittelwertvonAnzFlug
▶	1,3333333333333333

3.4.4 Umsatz pro Flug

```
SELECT
        tbl_Buchung.FlugNr,
        Sum(tbl_Klasse.Preis) AS SummevonPreis
FROM
        tbl_Klasse INNER JOIN tbl_Buchung ON tbl_Klasse.KlasseNr =
        tbl_Buchung.KlasseNr
GROUP BY
        tbl_Buchung.FlugNr;
```

	FlugNr	SummevonPreis
▶	1	100
	2	51
	3	100

3.4.5 Welche Flughäfen werden von welchen Fluggesellschaften angeflogen?

```
SELECT
        tbl_Flughafen.Name,
        tbl_Fluggesellschaft.Name
FROM
        (tbl_Fluggesellschaft INNER JOIN tbl_Flug ON
        tbl_Fluggesellschaft.FluggesellschaftNr = tbl_Flug.FluggesellschaftNr) INNER
        JOIN (tbl_Flughafen INNER JOIN tbl_Flugstationen ON
        tbl_Flughafen.FlughafenNr = tbl_Flugstationen.FlughafenNr) ON
        tbl_Flug.FlugNr = tbl_Flugstationen.FlugNr
GROUP
        BY tbl_Flughafen.Name, tbl_Fluggesellschaft.Name
ORDER BY
        tbl_Flughafen.Name;
```

	tbl_Flughafen.Name	tbl_Fluggesellschaft.Name
▶	berlin	condor
	berlin	lufthansa
	ffm	condor
	ffm	lufthansa
	hamburg	lufthansa
	münchen	condor
	münchen	lufthansa

4. QS-Abwicklung

4.1 Aufgabenstellung

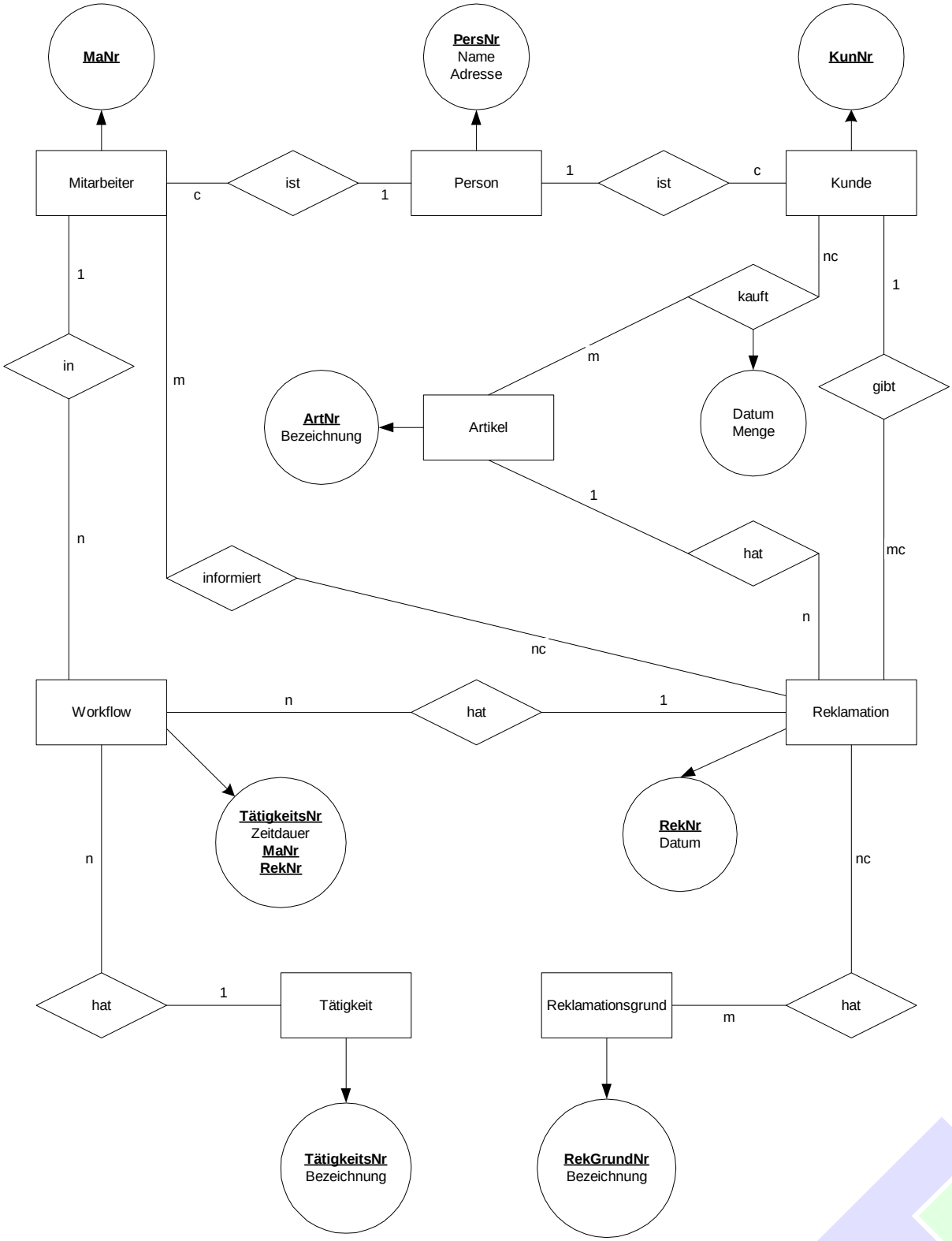
Für die Abwicklung und Verfolgung von Kundenreklamationen soll eine Datenbank entworfen werden. Notwendige Eintragungen in die Datenbank sollen sicherstellen, dass zukünftig die Fertigprodukt-Qualität verbessert wird. Im Einzelnen werden folgende Anforderungen an die Datenbank gestellt:

- Speicherung aller Kundenreklamationen in Form eines Ticketsystems. Ticketsystem bedeutet, dass jede Reklamation eine eindeutige Nummer erhält.
- Klassifizierung des Reklamationsgrundes
- Zuordnung der Reklamation auf eine Kundennummer, betroffenes Produkt (Artikel) und auf den Kundenauftrag.
- Erstellung von einem Informationsverteiler, d.h. mit der Reklamation soll es möglich sein, verschiedene Mitarbeiter des Unternehmens von der Reklamation zu unterrichten.
- Erstellen eines Workflows für die Abarbeitung der Reklamation. In dem Workflow wird festgelegt, welche Tätigkeiten für die erfolgreiche Abarbeitung der Reklamation notwendig sind und welche Personen die Workflowschritte ausführen müssen.

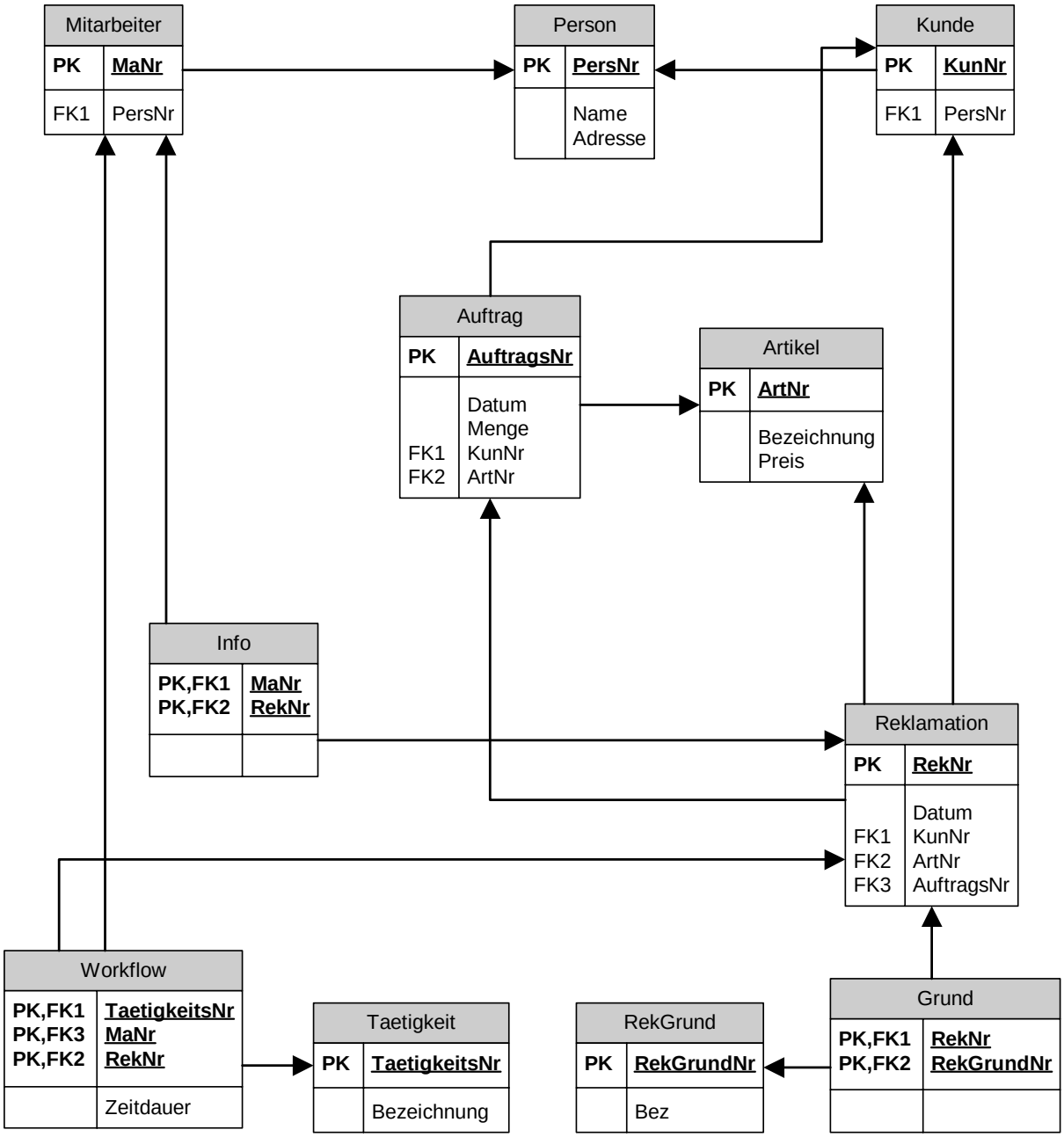
Häufige Auswertungen sind z.B.:

- Wie häufig kam es zur Reklamation bei einem Produkt bzw. bei einem Kunden?
- Welche Gründe gab der Kunde für die Reklamation an?
- Welche Aufgaben bzw. Informationen bekommt ein bestimmter Mitarbeiter?
- Welche Aufgaben muss ein Mitarbeiter erledigen und bis wann?
- Wie viele Tickets gab es innerhalb der letzten 12 Monate?

4.2 Entity-Relationship-Diagramm



4.3 Logisches Datenbankkonzept



4.4 SQL-Befehle (MaxDB-SQL)

4.4.1 Wie häufig kam es zur Reklamation bei einem Produkt bzw. bei einem Kunden?

Produkt:

```
SELECT
    artikel.artnr,
    COUNT(reklamation.reknr) AS reklamationen
FROM
    reklamation INNER JOIN artikel ON artikel.artnr=reklamation.artnr
GROUP BY
    artikel.artnr
```

	ARTNR	REKLAMATIONEN
1	1	3
2	2	1
3	3	2
4	4	2

Kunde:

```
SELECT
    kunde.kunnr,
    COUNT(reklamation.reknr) AS reklamationen
FROM
    reklamation INNER JOIN kunde ON kunde.kunnr=reklamation.kunnr
GROUP BY
    kunde.kunnr
```

	KUNNR	REKLAMATIONEN
1	1	2
2	2	2
3	4	3
4	5	1

4.4.2 Welche Gründe gab der Kunde für die Reklamation an?

```
SELECT
      reklamation.reknr,
      rekgrund.bez AS reklamationsgrund
FROM
      reklamation INNER JOIN grund ON reklamation.reknr=grund.reknr
      INNER JOIN rekgrund ON grund.rekgrundnr=rekgrund.rekgrundnr
```

	REKNR	REKLAMATIONSGRUND
1	1	a
2	1	c
3	2	f
4	3	c
5	4	b
6	4	e
7	4	g
8	5	g
9	6	c
10	7	c
11	7	d
12	8	a
13	8	b
14	8	f

4.4.3 Welche Aufgaben bzw. Informationen bekommt ein bestimmter Mitarbeiter?

Aufgaben:

```
SELECT
      mitarbeiter.manr,
      taetigkeit.bezeichnung
FROM
      mitarbeiter INNER JOIN workflow ON mitarbeiter.manr=workflow.manr INNER
      JOIN taetigkeit ON taetigkeit.taetigkeitsnr=workflow.taetigkeitsnr
ORDER BY mitarbeiter.manr
```

	MANR	BEZEICHNUNG
1	1	a
2	1	c
3	2	c
4	3	c
5	3	f
6	4	b
7	4	e
8	6	b

Informationen:

```
SELECT
      mitarbeiter.manr,
      info.reknr
FROM
      mitarbeiter INNER JOIN info ON mitarbeiter.manr=info.manr
ORDER BY mitarbeiter.manr
```

	MANR	REKNR
1	1	1
2	1	2
3	1	3
4	1	4
5	1	5
6	1	6
7	1	7
8	1	8

4.4.4 Welche Aufgaben muss ein Mitarbeiter erledigen und bis wann?

```
SELECT
      mitarbeiter.manr,
      taetigkeit.bezeichnung,
      ADDDATE(reklamation.datum,workflow.zeitdauer) AS TaetigkeitEnde
FROM
      mitarbeiter INNER JOIN workflow ON mitarbeiter.manr=workflow.manr INNER
      JOIN taetigkeit ON  taetigkeit.taetigkeitsnr=workflow.taetigkeitsnr INNER JOIN
      reklamation ON workflow.reknr=reklamation.reknr
ORDER BY mitarbeiter.manr
```

	MANR	BEZEICHNUNG	TAETIGKEITENDE
1	1	a	2005-05-02
2	1	c	2005-05-04
3	2	c	2005-05-04
4	3	c	2005-05-04
5	3	f	2005-05-04
6	4	b	2005-05-06
7	4	e	2005-05-05
8	6	b	2005-05-04

4.4.5 Wie viele Tickets gab es innerhalb der letzten 12 Monate?

```
SELECT
        COUNT(reknr) AS AnzahlTickets
FROM
        reklamation
WHERE
        datum > SUBDATE(now(),365)
```

	ANZAHLTICKETS
1	8

5. Produktionsverfolgung bei sicherheitsrelevanten Teilen

5.1 Aufgabenstellung

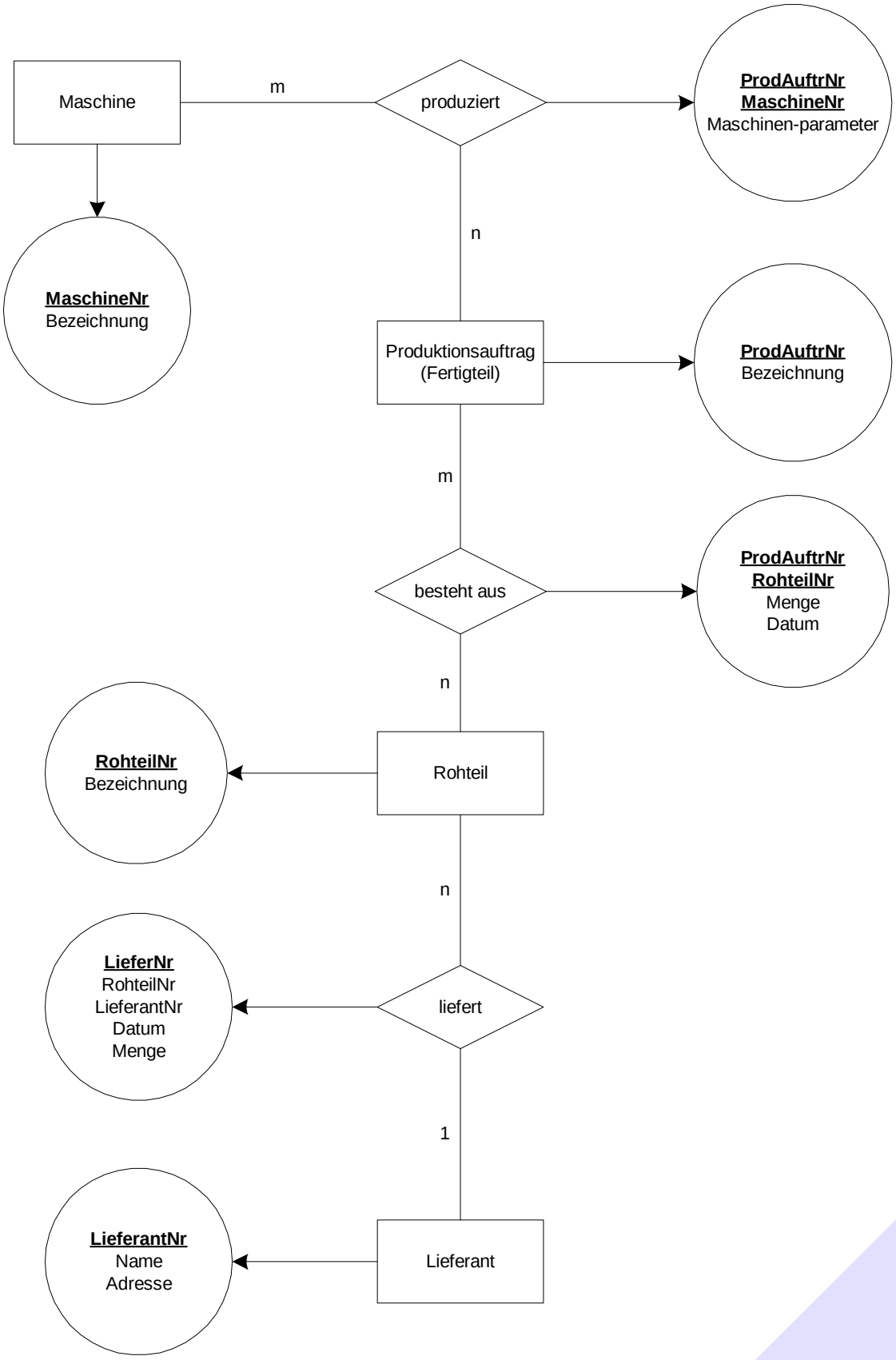
Für die Verfolgung der Produktion sicherheitsrelevanter Teile ist es notwendig, alle die Produktion betreffenden Informationen in einer Datenbank zu hinterlegen. Solche Informationen sind im Einzelnen:

- Speicherung der Rohteile (z.B. Schrauben, Metallteile, Kunststoffteile usw.) und Fertigteile (z.B. Airbags, Bremsblock)
- Nachweis aller in der Produktion verwendeten Rohteile insbesondere Informationen zu folgenden Sachverhalten:
  - Welcher Lieferant hat ein Rohteil, in welcher Menge, an welchem Tag geliefert? Eindeutige Kennzeichnung der einzelnen Wareneingänge.
  - Welche Rohteile aus welchem Wareneingang werden für die Produktion eines Fertigteils verwendet und an welchem Tag?
- Produktionsverfahren
- Erstellen von eindeutigen Produktionsaufträgen. Wichtige Attribute des Produktionsauftrags sind z.B. Produktionsmenge, Fertigteil, Menge und Datum der Produktion.
- Nachweis der Maschinen, die an der Produktion beteiligt waren. Des Weiteren soll pro Maschine und gefertigtem Produkt erkennbar sein, welche Maschinenparameter (Temperatur, Geschwindigkeit, Pressdruck, Luftfeuchtigkeit usw.) während des Produktionsvorgangs eingestellt waren.

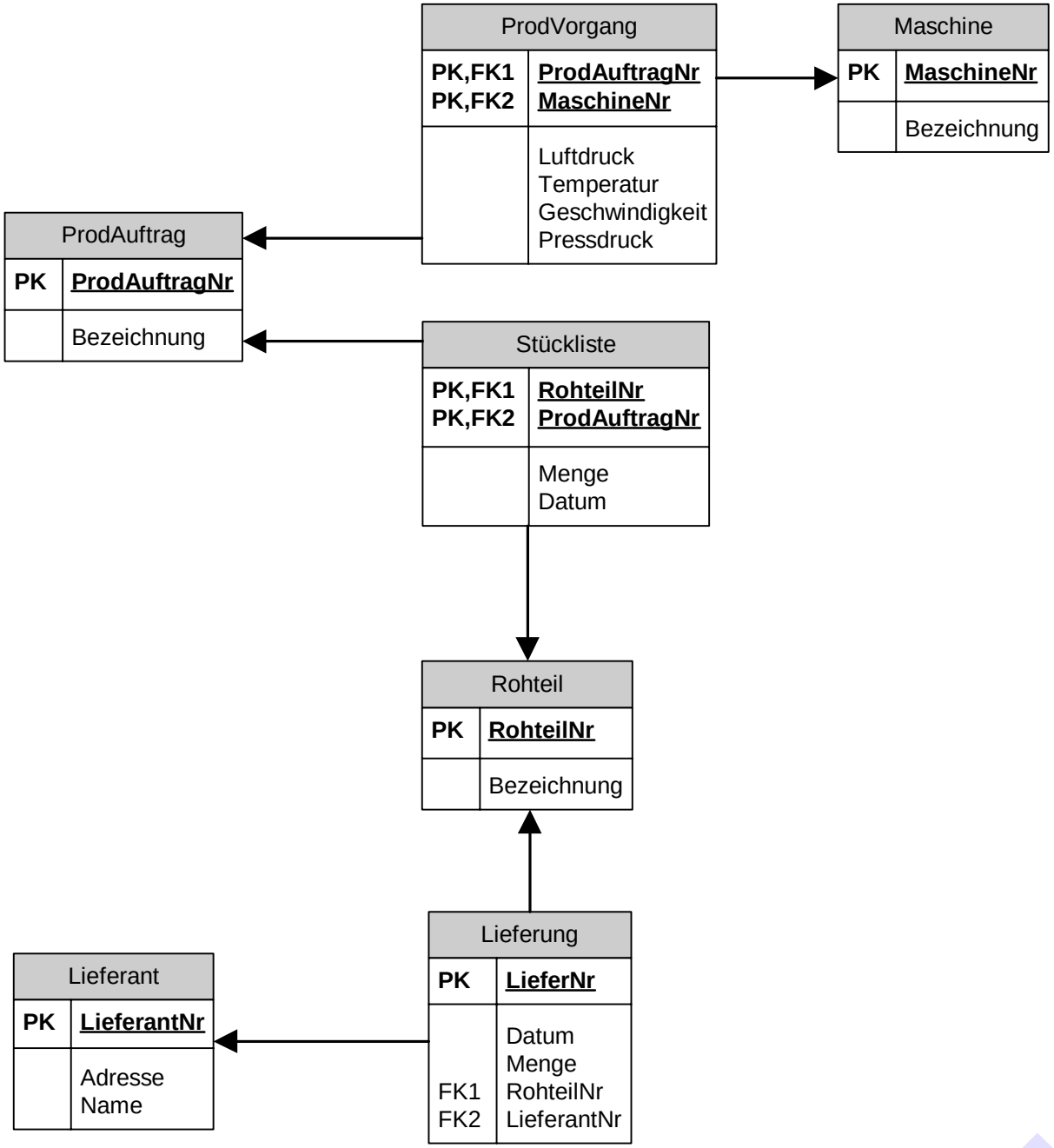
Häufige Auswertungen sind z.B.:

- Welcher Lieferant hat einen Rohteil geliefert und an welchen Tagen?
- Welche Rohteile von welchem Lieferanten sind in einem Produktionsauftrag verwendet worden?
- Unter welchen Produktionsbedingungen wurden die Fertigteile produziert.
- Wie viele Fertigprodukte bzw. Rohteile wurden innerhalb eines definierten Zeitraums verbraucht?

5.2 Entity-Relationship-Diagramm



5.3 Logisches Datenbankkonzept



5.4 SQL-Befehle (MS-SQL – Access)

5.4.1 Welcher Lieferant hat einen Rohteil geliefert und an welchen Tagen?

```
SELECT
        Lieferant.Name AS Lieferant,
        Lieferung.Datum AS Lieferdatum
FROM
        Lieferung INNER JOIN Lieferant ON Lieferung.LieferantNr =
        Lieferant.LieferantNr
WHERE
        RohteilNr = 111;
```

Lieferant	Lieferdatum
Fabrik01	01.01.2004
Fabrik01	05.01.2004
Fabrik02	06.01.2004
Fabrik03	03.01.2004

5.4.2 Welche Rohteile von welchem Lieferanten sind in einem Produktionsauftrag verwendet worden?

```
SELECT
        Lieferant.Name,
        ProdAuftrag.Bezeichnung,
        Rohteil.Bezeichnung
FROM
        ((((Lieferant INNER JOIN Lieferung ON Lieferant.LieferantNr =
        Lieferung.LieferantNr) INNER JOIN Rohteil ON Lieferung.RohteilNr =
        Rohteil.RohteilNr) INNER JOIN Stückliste ON Rohteil.RohteilNr =
        Stückliste.RohteilNr) INNER JOIN ProdAuftrag ON ProdAuftrag.ProdAuftragNr
        = Stückliste.ProdAuftragNr)
WHERE
        ProdAuftrag.ProdAuftragNr = 1234;
```

Name	ProdAuftrag.Bezeichnung	Rohteil.Bezeichnung
Fabrik01	Fertigung01	Schraube
Fabrik03	Fertigung01	Schraube
Fabrik01	Fertigung01	Schraube
Fabrik02	Fertigung01	Schraube
Fabrik02	Fertigung01	Metallteil
Fabrik02	Fertigung01	Metallteil
Fabrik03	Fertigung01	Metallteil

5.4.3 Unter welchen Produktionsbedingungen wurden die Fertigteile produziert.

```
SELECT
        ProdAuftrag.Bezeichnung AS Fertigteil,
        ProdVorgang.Temperatur,
        ProdVorgang.Luftdruck,
        ProdVorgang.Geschwindigkeit,
        ProdVorgang.Pressdruck
FROM
        ProdVorgang INNER JOIN ProdAuftrag ON
        ProdVorgang.ProdAuftragNr = ProdAuftrag.ProdAuftragNr
ORDER BY
        ProdVorgang.ProdAuftragNr;
```

	Fertigteil	Temperatur	Luftdruck	Geschwindigk	Pressdruck
	Fertigung01	15	45	30	30
	Fertigung01	10	20	20	5
	Fertigung01	45	10	10	45
	Fertigung01	90	40	10	35
	Fertigung02	10	35	28	34
	Fertigung02	40	10	24	7
	Fertigung02	30	45	34	10
	Fertigung02	85	20	15	23

5.4.4 Wie viele Fertigprodukte bzw. Rohteile wurden innerhalb eines definierten Zeitraums verbraucht?

```
SELECT
        SUM(Stückliste.Menge) AS Gesamtmenge
FROM
        Stückliste
WHERE
        Stückliste.Datum BETWEEN #2004-01-10# AND #2004-01-15#;
```

	Gesamtmenge
▶	5

6. Stammdaten-Verwaltung für Produktionsartikel

6.1 Aufgabenstellung

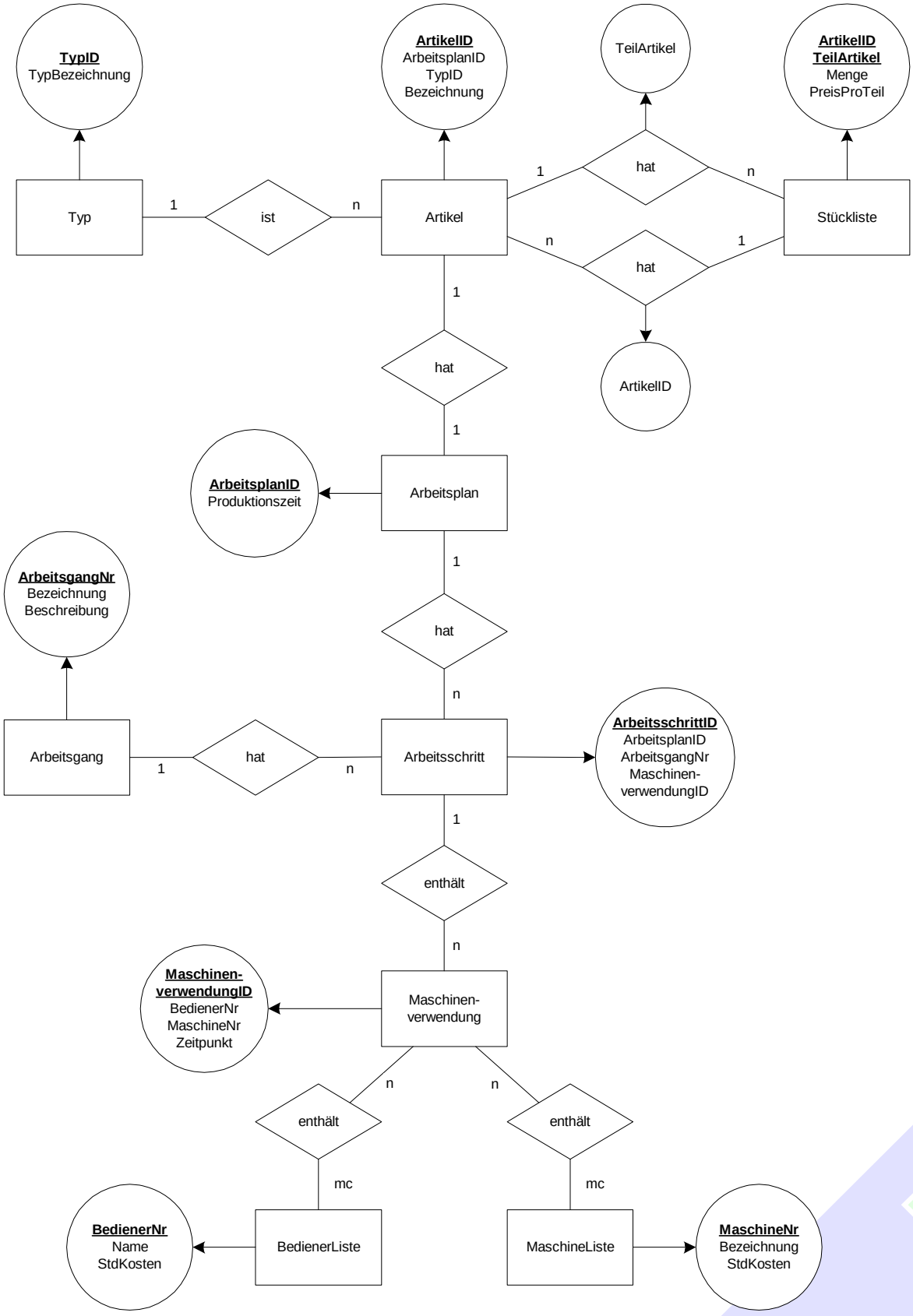
Das Produktionsunternehmen „Schnellgut“ will zukünftig die Daten der Fertigung in einem Datenbanksystem speichern. Als Anforderungen werden definiert:

- Alle Teile, Baugruppen und Fertigartikel sollen in einer gemeinsamen Artikeltabelle gehalten werden.
- Zu jedem Artikel ist eine Stückliste und ein Arbeitsplan anzulegen. In der Stückliste werden alle Artikel geführt, die für den Zusammenbau benötigt werden. Der Arbeitsplan wiederum enthält die Arbeitsfolgen, die notwendig sind, um die Teile einer Stückliste zu einem Fertigteil zusammenzubauen.
- In der Stücklistenposition sollen als Mindestinformationen folgende Felder gespeichert werden: verbaute Teile/Baugruppen, die Menge und der Preis eines Teils.
- Im Arbeitsplan ist die Speicherung folgender Felder notwendig:
  - Maschine und Maschinenbezeichnung (incl. Kosten pro Std.),
  - Maschinenbediener mit Namen und Kosten pro Stunde,
  - Produktionszeit,
  - Standardarbeitsgang mit Bezeichnung (Der Standardarbeitsgang soll eine allgemeine Typisierung der Arbeitsfolge erlauben, z.B. Montieren)

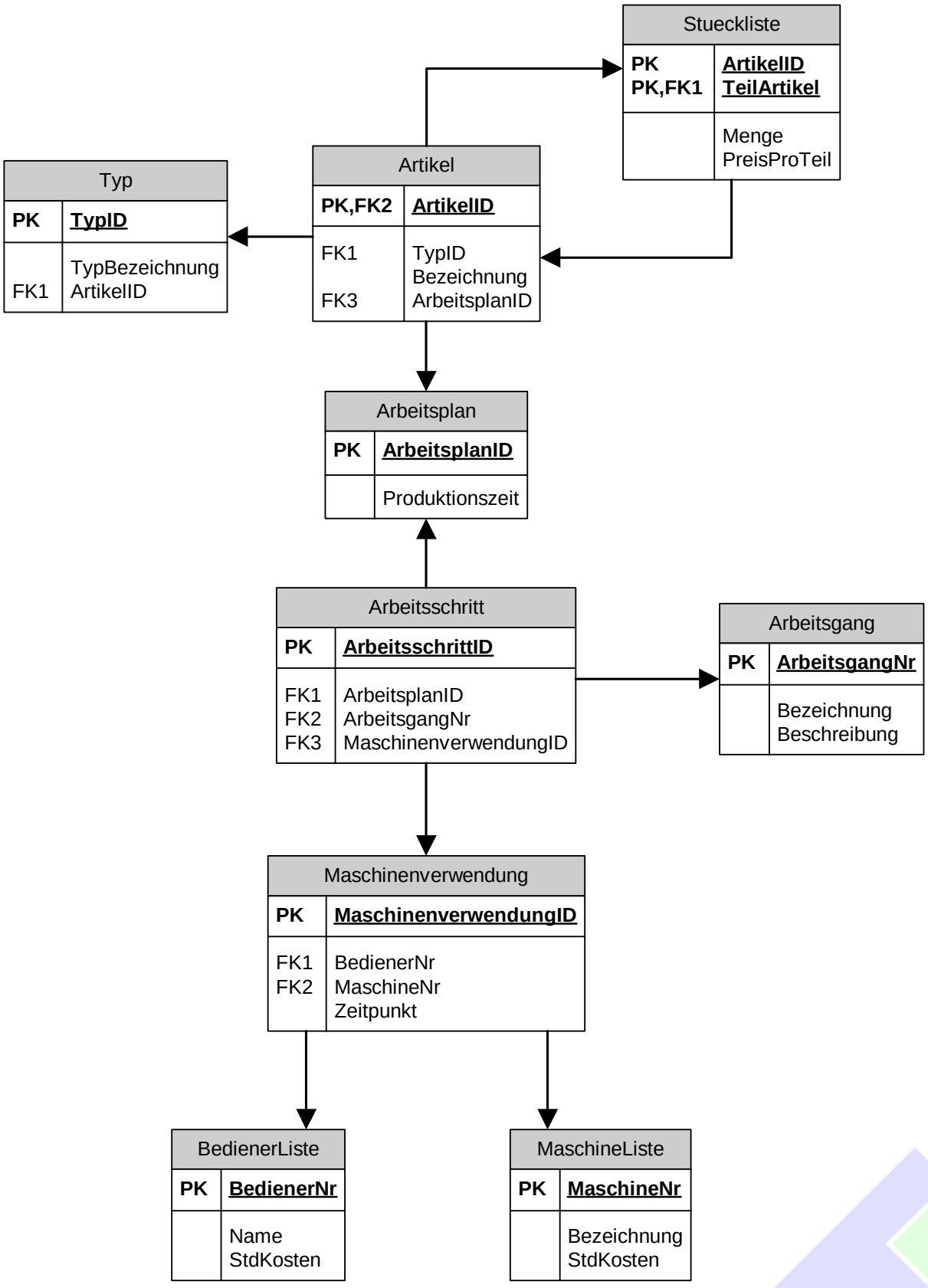
Häufige Auswertungen sind z.B.:

- Erstellen einer Teileverwendung d.h. welches Teil wird in welchem Fertigteil verbaut?
- Welche Maschine wird für welches Fertigteil benötigt?
- Welcher Standardarbeitsgang wird wo verwendet?
- Welcher Mitarbeiter (Maschinenbediener) arbeitet an welchem Arbeitsgang?
- Wie hoch ist der Materialpreis bzw. Arbeitsplanpreis?

6.2 Entity-Relationship-Diagramm



6.3 Logisches Datenbankkonzept



6.4 SQL-Befehle (MS-SQL – Access)

6.4.1 Erstellen einer Teileverwendung d.h. welches Teil wird in welchem Fertigteil verbaut?

```
SELECT
        Typ.TypID,
        Typ.TypBezeichnung,
        Stueckliste.TeilArtikel,
        Artikel.ArtikelID,
        Artikel.Bezeichnung
FROM
        (Stueckliste INNER JOIN Artikel ON Artikel.ArtikelID=Stueckliste.ArtikelID)
        INNER JOIN Typ ON Typ.TypID=Artikel.TypID
WHERE
        Typ.TypID = 1
ORDER BY
        Stueckliste.TeilArtikel;
```

	TypID	TypBezeichnun	TeilArtikel	ArtikelID	Bezeichnung
▶	1	Fertigteil	2	5	Motor
	1	Fertigteil	3	7	Bleiguss
	1	Fertigteil	3	5	Motor
*	(AutoWert)			(AutoWert)	

6.4.2 Welche Maschine wird für welches Fertigteil benötigt?

```
SELECT
        MaschineListe.Bezeichnung as Maschine,
        Artikel.ArtikelID,
        Typ.TypID,
        Typ.TypBezeichnung
FROM
        Typ INNER JOIN ((Arbeitsplan INNER JOIN ((MaschineListe INNER JOIN
        Maschinenverwendung ON MaschineListe.MaschineNr =
        Maschinenverwendung.MaschineNr) INNER JOIN Arbeitsschritt ON
        Maschinenverwendung.MaschinenverwendungID =
        Arbeitsschritt.MaschinenverwendungID) ON Arbeitsplan.ArbeitsplanID =
        Arbeitsschritt.ArbeitsplanID) INNER JOIN Artikel ON Arbeitsplan.ArbeitsplanID
        = Artikel.ArbeitsplanID) ON Typ.TypID = Artikel.TypID
WHERE
        (((Typ.TypID)=1));
```

	Maschine	ArtikelID	TypID	TypBezeichnun
▶	Presse	8	1	Fertigteil
	Förderband	7	1	Fertigteil

6.4.3 Welcher Standardarbeitsgang wird wo verwendet?

```
SELECT
        Arbeitsgang.ArbeitsgangNr,
        Arbeitsgang.Bezeichnung,
        Arbeitsplan.ArbeitsplanID
FROM
        Arbeitsplan INNER JOIN (Arbeitsgang INNER JOIN Arbeitsschritt ON
        Arbeitsgang.ArbeitsgangNr = Arbeitsschritt.ArbeitsgangNr) ON
        Arbeitsplan.ArbeitsplanID = Arbeitsschritt.ArbeitsplanID
ORDER BY
        Arbeitsgang.ArbeitsgangNr;
```

	ArbeitsgangNr	Bezeichnung	ArbeitsplanID
▶	1	Schrauben	2
	2	Montieren	1
*	(AutoWert)		(AutoWert)

6.4.4 Welcher Mitarbeiter (Maschinenbediener) arbeitet an welchem Arbeitsgang?

```
SELECT
        BedienerListe.BedienerNr,
        BedienerListe.Name,
        Arbeitsgang.ArbeitsgangNr,
        Arbeitsgang.Bezeichnung
FROM
        Arbeitsgang INNER JOIN ((BedienerListe INNER JOIN Maschinenverwendung
        ON BedienerListe.BedienerNr = Maschinenverwendung.BedienerNr) INNER
        JOIN Arbeitsschritt ON Maschinenverwendung.MaschinenverwendungID =
        Arbeitsschritt.MaschinenverwendungID) ON Arbeitsgang.ArbeitsgangNr =
        Arbeitsschritt.ArbeitsgangNr;
```

	BedienerNr	Name	ArbeitsgangNr	Bezeichnung
▶	1	Hemel	2	Montieren
	2	Lipinski	1	Schrauben
*	(AutoWert)		(AutoWert)	

6.4.5 Wie hoch ist der Materialpreis bzw. Arbeitsplanpreis?

Materialpreis:

```
SELECT
        Stueckliste.ArtikelID,
        Stueckliste.Menge*Stueckliste.PreisProTeil AS Materialpreis
FROM
        Stueckliste;
```

ArtikelID	Materialpreis
1	100
2	40
0	

Arbeitsplanpreis:

```
SELECT
        Artikel.ArtikelID,
        Arbeitsplan.ArbeitsplanID,
        MaschineGK+BedienerGK AS GesamtKosten
FROM
        ((Arbeitsplan INNER JOIN
                [SELECT
                        Arbeitsplan.ArbeitsplanID,
                        SUM(MaschineListe.StdKosten*Arbeitsplan.Produktionszeit) AS
                        MaschineGK
                FROM
                        ((Arbeitsplan INNER JOIN Arbeitsschritt ON
                        Arbeitsplan.ArbeitsplanID=Arbeitsschritt.ArbeitsplanID) INNER
                        JOIN Maschinenverwendung ON
                        Maschinenverwendung.MaschinenverwendungID=
                        Arbeitsschritt.MaschinenverwendungID) INNER JOIN
                        MaschineListe ON
                        MaschineListe.MaschineNr=Maschinenverwendung.MaschineNr
                GROUP BY
                        Arbeitsplan.ArbeitsplanID] AS MA
        ON Arbeitsplan.ArbeitsplanID = MA.ArbeitsplanID) INNER JOIN
                [SELECT
                        Arbeitsplan.ArbeitsplanID,
                        SUM (BedienerListe.StdKosten*Arbeitsplan.Produktionszeit)
                        AS BedienerGK
                FROM
                        ((Arbeitsplan INNER JOIN Arbeitsschritt ON
                        Arbeitsplan.ArbeitsplanID=Arbeitsschritt.ArbeitsplanID) INNER
                        JOIN Maschinenverwendung ON
                        Maschinenverwendung.MaschinenverwendungID=Arbeitsschritt.
                        MaschinenverwendungID) INNER JOIN BedienerListe ON
                        BedienerListe.BedienerNr=Maschinenverwendung.BedienerNr
                GROUP BY
                        Arbeitsplan.ArbeitsplanID
                ] AS BE
ON Arbeitsplan.ArbeitsplanID = BE.ArbeitsplanID) INNER JOIN Artikel
ON Arbeitsplan.ArbeitsplanID = Artikel.ArbeitsplanID;
```

	ArtikelID	ArbeitsplanID	GesamtKosten
▶	2	1	39
	3	2	55
	6	2	55