

Datenbanksysteme

Übungsbeispiel 2
Paul Staroch, 0425426
Datum: 28. November 2005
Erstellt mit L^AT_EX

1 Runde 1

1.1 Aufgabenstellung

Lösen Sie für das Beispiel folgende Aufgaben:

- Erstellen Sie für Ihre Angabe ein Entity-Relationship-Diagramm.
- Leiten Sie aus dem ER-Diagramm die Relationen der Datenbank in dritter Normalform so ab, dass sie bundtreu und abhängigkeittreu sind und implementieren Sie diese mittels SQL-Statements.
- Fügen Sie mittels INSERT-Statements Testdaten in Ihre Relationen ein.
- Für die Abgabe erstellen Sie ein Commandfile, das sämtliche für die Lösung der Aufgabenstellung benötigten SQL-Statements enthält.

1.2 Angabe

Eine Datenbank für eine Hausverwaltungsgesellschaft soll implementiert werden. Diese Gesellschaft besteht aus mehreren Verwaltungsstellen, die jeweils für bestimmte Häuser zuständig sind.

- Die einzelnen Verwaltungsstellen haben eine eindeutige ID, eine Adresse, sowie ein jährliches Budget. Sie sind jeweils für beliebig viele Häuser zuständig. Jede Verwaltungsstelle hat beliebig viele Mitarbeiter.
- Mitarbeiter arbeiten in genau einer Verwaltungsstelle und haben innerhalb dieser eine eindeutige Mitarbeiternummer, zusätzlich soll für sie der Name, die Adresse und das Gehalt gespeichert werden.
- Häuser werden durch Straße und Hausnummer identifiziert und haben ein Baujahr. Für jedes Haus ist genau eine Verwaltungsstelle zuständig.
- Häuser sind außerdem einer bestimmten Sanierungskategorie zugeordnet. Sanierungskategorien haben eine eindeutige Bezeichnung sowie einen Preis pro Quadratmeter. (Anmerkung: der Preis/m² wird später für die Berechnung der Betriebskosten der Wohnungen verwendet, d.h. die Sanierungsbedürftigkeit des jeweiligen Hauses ist ausschlaggebend für die Höhe der Betriebskosten!)
- Für Wohnungen ist die Anzahl der Quadratmeter und die Türnummer bekannt. Sie sind außerdem klarerweise genau einem Haus zugeordnet. (Anmerkung: es kann davon ausgegangen werden, dass die Türnummer innerhalb eines Hauses nur ein einziges Mal vorkommt!) Spezielle Wohnungen besitzen zusätzlich eine Kaufoption, für diese wird zusätzlich der Preis und die Laufzeit in Jahren gespeichert.
- Mieter haben eine Kontonummer und können eine oder mehrere Wohnungen mieten, dabei soll für jede gemietete Wohnung die Miete sowie das Vertragsdatum vermerkt sein. Eigentümer können ebenfalls mindestens eine oder beliebig viele Wohnungen besitzen, wobei für jede einzelne Wohnung das Kaufdatum bekannt ist. Umgekehrt können Wohnungen auch weder Eigentümer noch Mieter haben, d.h. nicht jede Wohnung ist entweder einem Eigentümer oder einem Mieter zugewiesen.
- Mieter und Eigentümer sind Personen. Personen haben eine eindeutige Sozialversicherungsnummer, einen Namen, einen Titel und ein Geschlecht. Personen zahlen Betriebskosten für jedes einzelne Haus, in dem sie Wohnungen besitzen oder mieten. (Anmerkung: wenn Personen im selben Haus mehrere Wohnungen besitzen und/oder mieten, so zahlen sie die Summe aller anfälligen Betriebskosten an das jeweilige Haus!)
- Sanierungsarbeiten, die für ein bestimmtes Haus anfallen, werden mit einer eindeutigen Identifikationsnummer sowie einer allgemeinen Beschreibung gespeichert, wobei für jedes Haus beliebig viele Sanierungsarbeiten anfallen können. Sanierungsarbeiten sind entweder als Ausschreibung oder als Auftrag spezifiziert. Sanierungsaufträge haben zusätzlich einen Status und sind genau einer Firma erteilt, umgekehrt können Firmen natürlich auch

mehrere Aufträge erhalten. Sanierungsausschreibungen haben einen flag ('true' oder 'false'), der angibt ob die Ausschreibung bereits beendet wurde oder nicht. Firmen können für beliebig viele Ausschreibungen einen Kostenvoranschlag machen, d.h. für jede Ausschreibung sollen die Kostenvoranschläge der interessierten Firmen vermerkt werden. Firmen haben einen eindeutigen Namen und einen Firmensitz.

2 Runde 2

2.1 Aufgabenstellung

- Lesen Sie das Kapitel PL/SQL in den Übungsunterlagen.
- Ergänzen Sie Ihre Daten damit die Queries sinnvoll getestet werden können, d.h. unter anderem, dass jede Query auch ein Ergebnis liefern muss.
- Die bei den Angaben kursiv gesetzten Daten dürfen von Ihnen geändert (d.h. an Ihren Datenbestand angepasst) werden.
- Organisieren Sie die abgeleiteten Relationen in Cluster und legen Sie eventuell noch Indices an. Beziehen Sie sich dabei nur auf die angegebenen Abfragen und versuchen Sie deren Effizienz zu steigern.
- Definieren Sie die ermittelten Cluster und Indizes mit den SQL-Statements **CREATE CLUSTER** beziehungsweise **CREATE INDEX** und die Relationen mit dem SQL-Statement **CREATE TABLE** unter Beachtung der jeweiligen Constraints (**NOT NULL**, **PRIMARY KEY**, **FOREIGN KEY**). Lesen Sie dazu zuerst den Teil über interaktives SQL in den Arbeitsunterlagen und studieren Sie insbesondere die Definitionen der benötigten SQL-Kommandos. Verwenden Sie dabei nur die Datentypen **INTEGER**, **VARCHAR2** und **DATE**. Tragen Sie in jede Relation mit **INSERT** mindestens 6 Tupel ein. Versuchen Sie im Hinblick auf die zu formulierenden Queries möglichst aussagekräftige Testdaten zu finden.
- Geben Sie alle Queries mit **SELECT**-Statements aus. Anschließend geben Sie sämtliche Datenbankobjekte, die Sie erstellt haben, wieder frei. Dazu verwenden Sie **DROP CLUSTER**, **DROP INDEX**, **DROP TABLE** und **DROP VIEW**.
- Für die Abgabe erstellen Sie ein Commandfile, das sämtliche für die Lösung der Aufgabenstellung benötigten SQL-Statements enthält.

2.2 Angabe

Lösen Sie die folgenden Probleme mittels SQL:

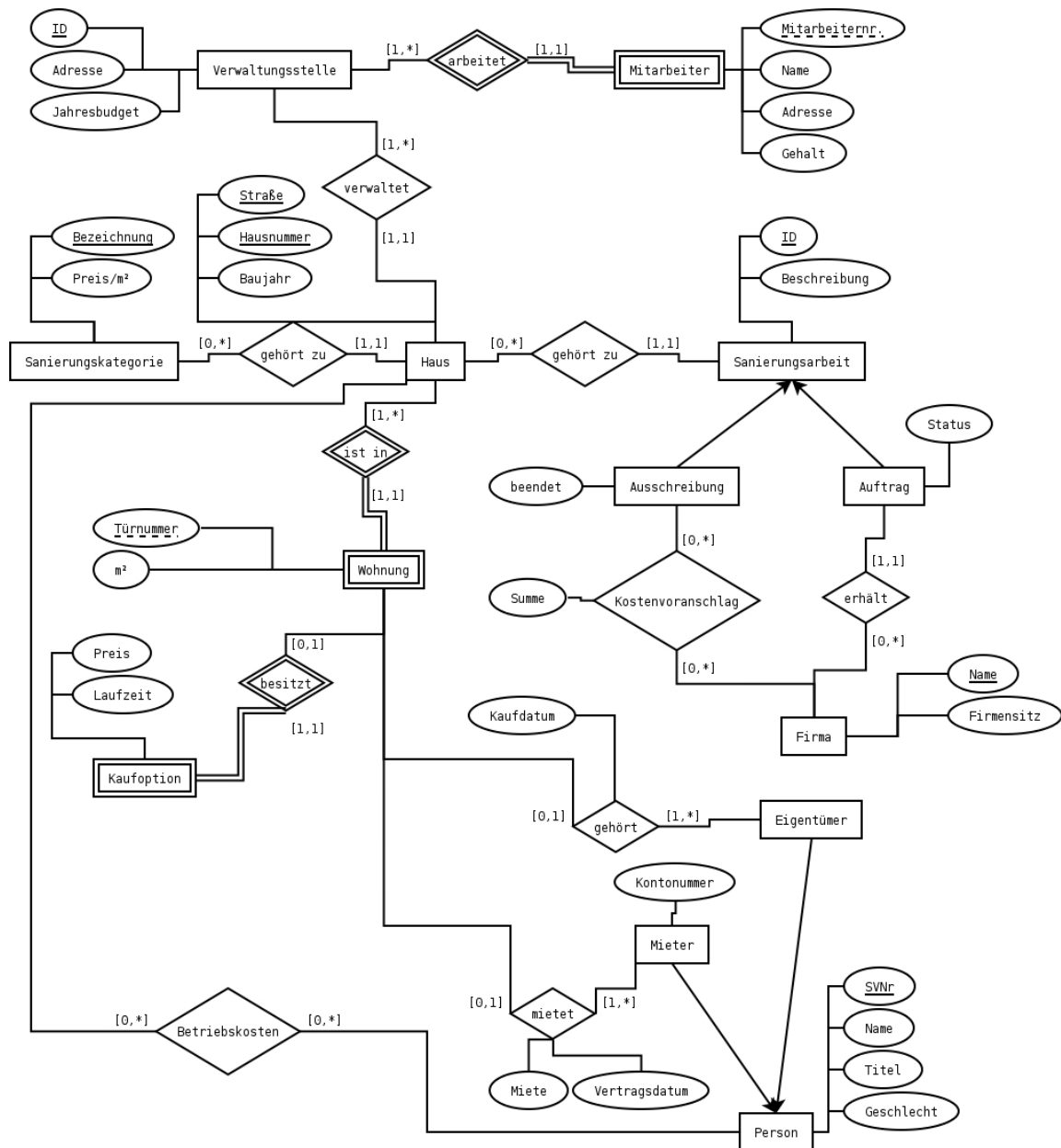
1. Geben Sie für alle Häuser die Anzahl der vermieteten Wohnungen und die Gesamtsumme deren Wohnflächen aus.
2. Geben Sie SVNr und Name jenes Eigentümers aus, der die meisten Wohnungen besitzt.
3. Geben Sie jene Häuser aus, bei denen sämtliche Sanierungsaufträge an unterschiedliche Firmen erteilt wurden.
4. Geben Sie alle Verwaltungsstellen gemeinsam mit ihrem Budget und der Anzahl der Häuser aus, für die mindestens 2 Sanierungsausschreibungen oder mindestens 2 Sanierungsaufträge existieren.
5. Geben Sie jene Firma gemeinsam mit ihrem Firmensitz aus, die die wenigsten Kostenvoranschläge gemacht hat. Dabei sollen allerdings nur jene Firmen berücksichtigt werden, die bisher zumindest einen Auftrag erteilt bekommen haben.
6. Geben Sie für alle Paare von Firmen und Verwaltungsstellen die Anzahl der erteilten Aufträge aus. (Anmerkung: Achten Sie darauf, dass wirklich alle Paare ausgegeben werden, d.h. auch Paare, bei denen die jeweilige Verwaltungsstelle der jeweiligen Firma noch nie einen Auftrag erteilt hat. Achten Sie außerdem darauf, dass jedes Paar nur einmal ausgegeben werden soll!)

Lösen Sie die folgenden Probleme mittels PL/SQL:

1. Schreiben Sie eine Prozedur, die für alle Personen die an ein Haus zu zahlenden Betriebskosten berechnet und den Betrag in die Datenbank einträgt bzw. aktualisiert. (Anmerkung: Betriebskostenberechnung etc. siehe Erläuterungen in Beispiel1!)

2. Schreiben Sie eine Prozedur, die für Wohnungen mit Sofortkaufoption den Preis berechnet. Übergeben sie hierfür den Schlüssel der Wohnung als Parameter, prüfen Sie anschließend ob die „übergebene“ Wohnung überhaupt eine Kaufoption besitzt. Ist dies nicht der Fall, ändern sie die Textmeldung der Oracle Exception „NO_DATA_FOUND“ auf folgenden Text: „Diese Wohnung besitzt keine Kaufoption!“. Andernfalls berechnen Sie den Verkaufspreis, indem Sie alle bisher bezahlten Mieten vom eigentlichen Preis abziehen, dabei muss allerdings noch auf die Befristung geachtet werden, d.h. ist eine Kaufoption beispielsweise auf 3 Jahre befristet, können maximal 36 Monatsmieten abgezogen werden. Geben Sie anschließend den Preis der Wohnung aus!
(Anmerkung: der Einfachheit halber berechnen Sie die Anzahl der bezahlten Mieten mit folgender Formel: (SYS-DATE - VertragsDatum)/31)
3. Schreiben Sie einen Trigger, der jedesmal ausgelöst wird, nachdem ein Update auf die Tabelle Ausschreibung vollzogen wurde. Prüfen Sie ob im Zuge dieses Updates die Ausschreibung beendet wurde (von 'false' auf 'true'). Falls ja, soll der Trigger einen der oder den besten Kostenvoranschlag auswählen und falls dessen Höhe das noch verfügbare Budget der jeweiligen Verwaltungsstelle nicht überschreitet, den Auftrag an die betroffenen Firma erteilen. Um den Auftrag zu erteilen, schreiben Sie auch eine Funktion die hierfür die kleinste freie SanierungsID berechnet und returniert. Anschließend soll das verfügbare jährliche Budget der Verwaltungsstelle noch um diesen Betrag verringert werden. Sollte das Budget nicht reichen, werfen Sie eine eigene Exception, die folgende Fehlermeldung ausgibt: „Budget zu gering!“
(Tipp: erstellen Sie am besten eine Prozedur (stored procedure) die all diese Aufgaben übernimmt und einen Trigger der lediglich diese Prozedur mit den benötigten Parametern (IDNr) aufruft!)

3 ER-Diagramm



4 Umwandlung in Relationen

Verwaltungsstelle	: {[ID : integer, Adresse : string, Jahresbudget : float]}
Mitarbeiter	: {[Verwaltungsstelle.ID : integer, Mitarbeiternummer : integer, Name : string, Adresse : string, Gehalt : float]}
Haus	: {[Straße : string, Hausnummer : integer, Baujahr : integer, Verwaltungsstelle.ID : integer, Sanierungskategorie.Bezeichnung : string]}
Sanierungskategorie	: {[Bezeichnung : string, Preis : float]}
Sanierungsarbeit	: {[ID : integer, Beschreibung : string, Haus.Straße : string, Haus.Hausnummer : string]}
Ausschreibung	: {[Sanierungsarbeit.ID : integer, beendet : boolean]}
Auftrag	: {[Sanierungsarbeit.ID : integer, Status : string, Firma.Name : string]}
Firma	: {[Name : string, Firmensitz : string]}
Wohnung	: {[Haus.Straße : string, Haus.Hausnummer : integer, Türnummer : integer, Fläche : float]}
Kaufoption	: {[Haus.Straße : string, Haus.Hausnummer : integer, Türnummer : integer, Preis : float, Laufzeit : integer]}
Person	: {[SVNr : integer, Name : string, Titel : string, Geschlecht : string]}
Mieter	: {[Person.SVNr : integer, Kontonummer : integer]}
Eigentümer	: {[Person.SVNr : integer]}
Kostenvoranschlag	: {[Ausschreibung.ID : integer, Firma.Name : string, Summe : float]}
gehört	: {[Haus.Straße : string, Haus.Hausnummer : integer, Wohnung.Türnummer : integer, Eigentümer.SVNr : integer, Kaufdatum : date]}
mietet	: {[Haus.Straße : string, Haus.Hausnummer : integer, Wohnung.Türnummer : integer, Mieter.SVNr : integer, Vertragsdatum : date, Miete : float]}
Betriebskosten	: {[Haus.Straße : string, Haus.Hausnummer : integer, Person.SVNr : integer, Summe : float]}

5 (PL-/)SQL-Statements

Anmerkung: Sinnvollerweise wurde nur die Ausgabe der Abfragen in eine Datei umgeleitet, nicht jedoch das Erstellen der Datenbankstrukturen, das Einfügen der Testdaten sowie das Löschen der Datenbankstrukturen.

5.1 Erstellen der Cluster, Tabellen und Indizes

```

/**
 * Datenbanksysteme, WS 2005, Beispiel 2
 * Paul Staroch, 0425426
 * CREATE.SQL – Erstellen von Clustern, Relationen, Indizes
 */

— Systemspezifische Einstellungen ueberschreiben
ALTER SESSION SET NLS_DATE_FORMAT='dd-Mon-yy';
ALTER SESSION SET NLS_DATE_LANGUAGE=english;

— Cluster

CREATE CLUSTER Haeuser (
    Strasse VARCHAR2(100),
    Hausnummer INTEGER
);

CREATE CLUSTER Wohnungen (
    Strasse VARCHAR2(100),
    Hausnummer INTEGER,
    Tuernummer INTEGER
);

CREATE CLUSTER Arbeiten (
    Ausschreibung INTEGER
);

CREATE CLUSTER Personen (
    SVNr INTEGER
);

```

— Relationen und Indizes

```

CREATE TABLE Verwaltungsstelle (
    ID                INTEGER          NOT NULL,
    Adresse           VARCHAR2(100)    NOT NULL,
    Jahresbudget      FLOAT            NOT NULL,
    PRIMARY KEY (ID)
);

CREATE TABLE Mitarbeiter (
    Verwaltungsstelle INTEGER          NOT NULL,
    Mitarbeiternummer INTEGER          NOT NULL,
    Name              VARCHAR2(100)    NOT NULL,
    Gehalt             FLOAT            NOT NULL,
    PRIMARY KEY (Verwaltungsstelle, Mitarbeiternummer),
    FOREIGN KEY (Verwaltungsstelle) REFERENCES Verwaltungsstelle (ID)
);

CREATE TABLE Sanierungskategorie (
    Bezeichnung       VARCHAR2(100)    NOT NULL,
    Preis             FLOAT            NOT NULL,
    PRIMARY KEY (Bezeichnung)
);

CREATE TABLE Haus (
    Strasse           VARCHAR2(100)    NOT NULL,
    Hausnummer        INTEGER          NOT NULL,
    Baujahr           INTEGER          NOT NULL,
    Verwaltungsstelle INTEGER          NOT NULL,
    Bezeichnung       VARCHAR2(100)    NOT NULL,
    PRIMARY KEY (Strasse, Hausnummer),
    FOREIGN KEY (Verwaltungsstelle) REFERENCES Verwaltungsstelle (ID),
    FOREIGN KEY (Bezeichnung) REFERENCES Sanierungskategorie (Bezeichnung)
) CLUSTER Haeuser (Strasse, Hausnummer);

CREATE TABLE Sanierungsarbeit (
    ID                INTEGER          NOT NULL,
    Beschreibung       VARCHAR2(255)    NOT NULL,
    Strasse           VARCHAR2(100)    NOT NULL,
    Hausnummer        INTEGER          NOT NULL,
    PRIMARY KEY (ID),
    FOREIGN KEY (Strasse, Hausnummer) REFERENCES Haus (Strasse, Hausnummer)
) CLUSTER Haeuser (Strasse, Hausnummer);

CREATE TABLE Ausschreibung (
    ID                INTEGER          NOT NULL,
    beendet           INTEGER          NOT NULL,
    PRIMARY KEY (ID),
    FOREIGN KEY (ID) REFERENCES Sanierungsarbeit (ID)
) CLUSTER Arbeiten (ID);

CREATE TABLE Firma (
    Name              VARCHAR2(100)    NOT NULL,
    Firmensitz        VARCHAR2(100)    NOT NULL,
    PRIMARY KEY (Name)
);

CREATE TABLE Auftrag (
    ID                INTEGER          NOT NULL,
    Status            VARCHAR2(100)    NOT NULL,
    Firma             VARCHAR2(100)    NOT NULL,
    PRIMARY KEY (ID),
    FOREIGN KEY (ID) REFERENCES Sanierungsarbeit (ID),
    FOREIGN KEY (Firma) REFERENCES Firma (Name)
) CLUSTER Arbeiten (ID);

CREATE INDEX AuftragIdx ON Auftrag (Firma);

CREATE TABLE Wohnung (
    Strasse           VARCHAR2(100)    NOT NULL,
    Hausnummer        INTEGER          NOT NULL,
    Tuernummer        INTEGER          NOT NULL,
    Flaeche           FLOAT            NOT NULL,

```

```

        PRIMARY KEY (Strasse, Hausnummer, Tuernummer),
        FOREIGN KEY (Strasse, Hausnummer) REFERENCES Haus (Strasse, Hausnummer)
    ) CLUSTER Wohnungen (Strasse, Hausnummer, Tuernummer);

CREATE INDEX WohnungIdx ON Wohnung (Strasse, Hausnummer);

CREATE TABLE Kaufoption (
    Strasse      VARCHAR2(100)    NOT NULL,
    Hausnummer   INTEGER          NOT NULL,
    Tuernummer   INTEGER          NOT NULL,
    Preis        FLOAT            NOT NULL,
    Laufzeit     INTEGER          NOT NULL,
    PRIMARY KEY (Strasse, Hausnummer, Tuernummer),
    FOREIGN KEY (Strasse, Hausnummer, Tuernummer) REFERENCES Wohnung (Strasse, Hausnummer,
        Tuernummer)
    ) CLUSTER Wohnungen (Strasse, Hausnummer, Tuernummer);

CREATE TABLE Person (
    SVNr         INTEGER          NOT NULL,
    Name         VARCHAR2(100)    NOT NULL,
    Titel        VARCHAR2(20),
    Geschlecht   VARCHAR2(1)     NOT NULL,
    PRIMARY KEY (SVNr)
    ) CLUSTER Personen (SVNr);

CREATE TABLE Mieter (
    SVNr         INTEGER          NOT NULL,
    Kontonummer  INTEGER          NOT NULL,
    PRIMARY KEY (SVNr),
    FOREIGN KEY (SVNr) REFERENCES Person (SVNr)
    ) CLUSTER Personen (SVNr);

CREATE TABLE Eigentuermer (
    SVNr         INTEGER          NOT NULL,
    PRIMARY KEY (SVNr),
    FOREIGN KEY (SVNr) REFERENCES Person (SVNr)
    ) CLUSTER Personen (SVNr);

CREATE TABLE Kostenvoranschlag (
    Ausschreibung INTEGER          NOT NULL,
    Firma         VARCHAR2(100)    NOT NULL,
    Summe         FLOAT            NOT NULL,
    PRIMARY KEY (Ausschreibung, Firma),
    FOREIGN KEY (Ausschreibung) REFERENCES Ausschreibung (ID),
    FOREIGN KEY (Firma) REFERENCES Firma (Name)
    ) CLUSTER Arbeiten (Ausschreibung);

CREATE INDEX KostenvoranschlagIdx ON Kostenvoranschlag (Firma);

CREATE TABLE gehoert (
    Strasse      VARCHAR2(100)    NOT NULL,
    Hausnummer   INTEGER          NOT NULL,
    Tuernummer   INTEGER          NOT NULL,
    Eigentuermer INTEGER          NOT NULL,
    Kaufdatum    DATE             NOT NULL,
    PRIMARY KEY (Strasse, Hausnummer, Tuernummer),
    FOREIGN KEY (Strasse, Hausnummer, Tuernummer) REFERENCES Wohnung (Strasse, Hausnummer,
        Tuernummer),
    FOREIGN KEY (Eigentuermer) REFERENCES Eigentuermer (SVNr)
    ) CLUSTER Wohnungen (Strasse, Hausnummer, Tuernummer);

CREATE INDEX gehoertIdx ON gehoert (Eigentuermer);

CREATE TABLE mietet (
    Strasse      VARCHAR2(100)    NOT NULL,
    Hausnummer   INTEGER          NOT NULL,
    Tuernummer   INTEGER          NOT NULL,
    Mieter       INTEGER          NOT NULL,
    Miete        FLOAT            NOT NULL,
    Vertragsdatum DATE            NOT NULL,
    PRIMARY KEY (Strasse, Hausnummer, Tuernummer),
    FOREIGN KEY (Strasse, Hausnummer, Tuernummer) REFERENCES Wohnung (Strasse, Hausnummer,
        Tuernummer),
    FOREIGN KEY (Mieter) REFERENCES Mieter (SVNr)

```

```

) CLUSTER Wohnungen (Strasse, Hausnummer, Tuernummer);

CREATE INDEX mietetIdx ON mietet (Mieter);

CREATE TABLE Betriebskosten (
    Strasse          VARCHAR2(100) NOT NULL,
    Hausnummer       INTEGER       NOT NULL,
    Person           INTEGER       NOT NULL,
    Summe            FLOAT         NOT NULL,
    PRIMARY KEY (Person, Strasse, Hausnummer),
    FOREIGN KEY (Strasse, Hausnummer) REFERENCES Haus (Strasse, Hausnummer),
    FOREIGN KEY (Person) REFERENCES Person (SVNr)
) CLUSTER Haeuser (Strasse, Hausnummer);

CREATE INDEX BetriebskostenIdx ON Betriebskosten (Person);

-- Indizes fuer Cluster

CREATE INDEX PersonenIndex ON CLUSTER Personen;
CREATE INDEX ArbeitenIndex ON CLUSTER Arbeiten;
CREATE INDEX WohnungenIndex ON CLUSTER Wohnungen;
CREATE INDEX HaeuserIndex ON CLUSTER Haeuser;

```

5.2 Einfügen der Testdaten

```

INSERT INTO Verwaltungsstelle VALUES (1, 'Favoritenstrasse_10', 100000);
INSERT INTO Verwaltungsstelle VALUES (2, 'Wiedner_Hauptstrasse_27', 50000);
INSERT INTO Verwaltungsstelle VALUES (3, 'Argentinierstrasse_53', 172000);
INSERT INTO Verwaltungsstelle VALUES (4, 'Panigl_gasse_11', 20000);
INSERT INTO Verwaltungsstelle VALUES (5, 'Apfelgasse_1', 10000);
INSERT INTO Verwaltungsstelle VALUES (6, 'Frankenberggasse_4', 50000);

INSERT INTO Mitarbeiter VALUES (1, 1, 'Manfred_Maier', 1200);
INSERT INTO Mitarbeiter VALUES (1, 2, 'Angelika_Mueller', 1700);
INSERT INTO Mitarbeiter VALUES (2, 1, 'Walter_Wagner', 1400);
INSERT INTO Mitarbeiter VALUES (2, 2, 'Antonia_Bachler', 1600);
INSERT INTO Mitarbeiter VALUES (3, 1, 'Erwin_Ebner', 2200);
INSERT INTO Mitarbeiter VALUES (4, 1, 'Cornelia_Fuerst', 1400);
INSERT INTO Mitarbeiter VALUES (5, 1, 'Martin_Graf', 1500);
INSERT INTO Mitarbeiter VALUES (6, 1, 'Agnes_Lang', 2000);
INSERT INTO Mitarbeiter VALUES (6, 2, 'Veronika_Bauer', 1700);

INSERT INTO Sanierungskategorie VALUES ('nicht_bewohnbar', 5);
INSERT INTO Sanierungskategorie VALUES ('Abbruchhaus', 10);
INSERT INTO Sanierungskategorie VALUES ('baufaellig', 20);
INSERT INTO Sanierungskategorie VALUES ('renovierungsbeduerftig', 25);
INSERT INTO Sanierungskategorie VALUES ('kleine_Ausbesserungen_notwendig', 40);
INSERT INTO Sanierungskategorie VALUES ('ausgezeichneter_Zustand', 50);

INSERT INTO Haus VALUES ('Taubstummengasse', 4, 1977, 1, 'renovierungsbeduerftig');
INSERT INTO Haus VALUES ('Gusshausstrasse', 31, 1844, 1, 'Abbruchhaus');
INSERT INTO Haus VALUES ('Schleifmuehl_gasse', 8, 1999, 2, 'ausgezeichneter_Zustand');
INSERT INTO Haus VALUES ('Operngasse', 18, 1933, 2, 'kleine_Ausbesserungen_notwendig');
INSERT INTO Haus VALUES ('Margaretenstrasse', 7, 1950, 3, 'baufaellig');
INSERT INTO Haus VALUES ('Paulanergasse', 2, 1822, 3, 'renovierungsbeduerftig');
INSERT INTO Haus VALUES ('Waaggasse', 11, 1917, 4, 'nicht_bewohnbar');
INSERT INTO Haus VALUES ('Floragasse', 20, 2002, 4, 'ausgezeichneter_Zustand');

INSERT INTO Sanierungsarbeit VALUES (1, 'Erneuerung_der_elektrischen_Installationen', 'Taubstummengasse', 4);
INSERT INTO Sanierungsarbeit VALUES (2, 'Einbau_neuer_Fenster', 'Operngasse', 18);
INSERT INTO Sanierungsarbeit VALUES (3, 'Streichen_der_Hausfassade', 'Margaretenstrasse', 7);
INSERT INTO Sanierungsarbeit VALUES (4, 'Aufzugseinbau', 'Floragasse', 20);
INSERT INTO Sanierungsarbeit VALUES (5, 'Integrieren_der_Gang-WCs_in_die_Wohnungen', 'Waaggasse', 11);
INSERT INTO Sanierungsarbeit VALUES (6, 'Integrieren_der_Gang-WCs_in_die_Wohnungen', 'Margaretenstrasse', 7);
INSERT INTO Sanierungsarbeit VALUES (7, 'Aufzugseinbau', 'Taubstummengasse', 4);
INSERT INTO Sanierungsarbeit VALUES (8, 'Einbau_neuer_Fenster', 'Margaretenstrasse', 7);
INSERT INTO Sanierungsarbeit VALUES (9, 'Streichen_der_Hausfassade', 'Floragasse', 20);
INSERT INTO Sanierungsarbeit VALUES (10, 'Aufzugseinbau', 'Margaretenstrasse', 7);

```

```

INSERT INTO Sanierungsarbeit VALUES (11, 'Aufzugseinbau', 'Floragasse', 20);
INSERT INTO Sanierungsarbeit VALUES (12, 'Einbau_neuer_Fenster', 'Taubstummengasse', 4);

INSERT INTO Ausschreibung VALUES (2, 1);
INSERT INTO Ausschreibung VALUES (3, 0);
INSERT INTO Ausschreibung VALUES (5, 0);
INSERT INTO Ausschreibung VALUES (8, 1);
INSERT INTO Ausschreibung VALUES (9, 0);

INSERT INTO Firma VALUES ('Hausrenovierung_Mueller', 'Wien');
INSERT INTO Firma VALUES ('Installateur_Maier', 'Graz');
INSERT INTO Firma VALUES ('Fassadenerneuerung_Gruber', 'Innsbruck');
INSERT INTO Firma VALUES ('Hausrenovierung_Bacher', 'Linz');
INSERT INTO Firma VALUES ('Installateur_Schmidt', 'Salzburg');
INSERT INTO Firma VALUES ('Fassadenerneuerung_Bauer', 'Eisenstadt');

INSERT INTO Auftrag VALUES (1, 'in_Vorbereitung', 'Installateur_Maier');
INSERT INTO Auftrag VALUES (4, 'im_Dezember_abgeschlossen', 'Hausrenovierung_Mueller');
INSERT INTO Auftrag VALUES (6, 'in_Vorbereitung', 'Hausrenovierung_Mueller');
INSERT INTO Auftrag VALUES (7, 'fast_fertig', 'Hausrenovierung_Mueller');
INSERT INTO Auftrag VALUES (10, 'fast_fertig', 'Hausrenovierung_Mueller');
INSERT INTO Auftrag VALUES (11, 'im_Jaenner_abgeschlossen', 'Hausrenovierung_Bacher');
INSERT INTO Auftrag VALUES (12, 'in_Planung', 'Fassadenerneuerung_Bauer');

INSERT INTO Wohnung VALUES ('Taubstummengasse', 4, 11, 67);
INSERT INTO Wohnung VALUES ('Taubstummengasse', 4, 54, 52);
INSERT INTO Wohnung VALUES ('Gusshausstrasse', 31, 13, 78);
INSERT INTO Wohnung VALUES ('Gusshausstrasse', 31, 16, 46);
INSERT INTO Wohnung VALUES ('Schleifmuehlgasse', 8, 35, 102);
INSERT INTO Wohnung VALUES ('Schleifmuehlgasse', 8, 8, 114);
INSERT INTO Wohnung VALUES ('Operngasse', 18, 23, 77);
INSERT INTO Wohnung VALUES ('Operngasse', 18, 4, 92);
INSERT INTO Wohnung VALUES ('Margaretenstrasse', 7, 11, 88);
INSERT INTO Wohnung VALUES ('Margaretenstrasse', 7, 9, 75);
INSERT INTO Wohnung VALUES ('Paulanergasse', 2, 32, 80);
INSERT INTO Wohnung VALUES ('Paulanergasse', 2, 17, 72);
INSERT INTO Wohnung VALUES ('Waaggasse', 11, 49, 66);
INSERT INTO Wohnung VALUES ('Waaggasse', 11, 17, 49);
INSERT INTO Wohnung VALUES ('Floragasse', 20, 12, 31);
INSERT INTO Wohnung VALUES ('Floragasse', 20, 1, 67);

INSERT INTO Kaufoption VALUES ('Taubstummengasse', 4, 11, 100000, 2);
INSERT INTO Kaufoption VALUES ('Gusshausstrasse', 31, 13, 72500, 3);
INSERT INTO Kaufoption VALUES ('Operngasse', 18, 23, 36000, 1);
INSERT INTO Kaufoption VALUES ('Waaggasse', 11, 17, 400000, 1);
INSERT INTO Kaufoption VALUES ('Paulanergasse', 2, 32, 100000, 2);
INSERT INTO Kaufoption VALUES ('Floragasse', 20, 12, 50000, 5);

INSERT INTO Person VALUES (5644120977, 'Anneliese_Maier', 'Dipl.-Ing.', 'w');
INSERT INTO Person VALUES (2322010152, 'Martin_Gruber', '', 'm');
INSERT INTO Person VALUES (1234170367, 'Anastasia_Mueller', 'Mag.', 'w');
INSERT INTO Person VALUES (2144220972, 'Gertrude_Moser', '', 'w');
INSERT INTO Person VALUES (2354070763, 'Gerhard_Lechner', '', 'm');
INSERT INTO Person VALUES (1245040381, 'Franz_Becker', 'Bakk.', 'm');
INSERT INTO Person VALUES (7855110875, 'Anna_Pichler', 'Dr.', 'w');

INSERT INTO Mieter VALUES (5644120977, 571456496456);
INSERT INTO Mieter VALUES (1234170367, 156987523954);
INSERT INTO Mieter VALUES (1245040381, 235698832522);
INSERT INTO Mieter VALUES (7855110875, 728548345873);
INSERT INTO Mieter VALUES (2144220972, 352387452845);
INSERT INTO Mieter VALUES (2354070763, 537263845523);

INSERT INTO Eigentuermer VALUES (2322010152);
INSERT INTO Eigentuermer VALUES (2144220972);
INSERT INTO Eigentuermer VALUES (2354070763);
INSERT INTO Eigentuermer VALUES (7855110875);
INSERT INTO Eigentuermer VALUES (1234170367);
INSERT INTO Eigentuermer VALUES (1245040381);

INSERT INTO Kostenvoranschlag VALUES (2, 'Hausrenovierung_Mueller', 10000);
INSERT INTO Kostenvoranschlag VALUES (3, 'Hausrenovierung_Mueller', 25000);
INSERT INTO Kostenvoranschlag VALUES (5, 'Hausrenovierung_Mueller', 30000);
INSERT INTO Kostenvoranschlag VALUES (3, 'Fassadenerneuerung_Gruber', 25000);

```

```

INSERT INTO Kostenvoranschlag VALUES (8, 'Hausrenovierung_Bacher', 50000);

INSERT INTO gehoert VALUES ('Schleifmuehlgasse', 8, 8, 2322010152, '12-Feb-2002');
INSERT INTO gehoert VALUES ('Floragasse', 20, 12, 2144220972, '28-Jul-2001');
INSERT INTO gehoert VALUES ('Paulanergasse', 2, 17, 2354070763, '19-Mar-1999');
INSERT INTO gehoert VALUES ('Operngasse', 18, 4, 7855110875, '28-Dec-1977');
INSERT INTO gehoert VALUES ('Operngasse', 18, 23, 7855110875, '30-Sep-1985');
INSERT INTO gehoert VALUES ('Margaretenstrasse', 7, 9, 1234170367, '01-Feb-1992');

INSERT INTO mietet VALUES ('Floragasse', 20, 12, 5644120977, 453, '28-Nov-2002');
INSERT INTO mietet VALUES ('Taubstummengasse', 4, 11, 1245040381, 262, '19-Jun-1997');
INSERT INTO mietet VALUES ('Waaggasse', 11, 49, 1234170367, 457, '17-Aug-2004');
INSERT INTO mietet VALUES ('Margaretenstrasse', 7, 11, 1245040381, 1245, '02-Jan-2005');
INSERT INTO mietet VALUES ('Schleifmuehlgasse', 8, 35, 2144220972, 355, '22-Sep-2002');
INSERT INTO mietet VALUES ('Waaggasse', 11, 17, 2354070763, 515, '13-Jul-1999');

```

5.3 Durchführen der verlangten Abfragen

```

/**
 * Datenbanksysteme, WS 2005, Beispiel 2
 * Paul Staroch, 0425426
 * LOESUNG.SQL – Geforderte Problemlösungen in SQL bzw. PL/SQL
 */

— Einstellungen fuer sinnvolle Ausgabe in Ergebnisdatei
SET PAGESIZE 100;
SET LINESIZE 210;
SET NUMWIDTH 15;

SET SERVEROUTPUT ON;

SPOOL ergebnis;

— 1.

BEGIN    /* vor jeder Abfrage Meldung ausgeben, um welche Abfrage es sich handelt */
    DBMS_OUTPUT.PUT_LINE('1. Anzahl der vermieteten Wohnungen und Summe derer Wohnflaechen aller
                          Haeuser');
END;
/

CREATE VIEW view1 AS
    SELECT w.Strasse, w.Hausnummer, COUNT(*) AS Anzahl, SUM(Flaechen) AS Flaechen
    FROM Wohnung w, mietet m
    WHERE w.Hausnummer=m.Hausnummer AND
          w.Strasse=m.Strasse AND
          w.Tuernummer=m.Tuernummer
    GROUP BY w.Strasse, w.Hausnummer;

CREATE VIEW view2 AS
    SELECT h.Strasse, h.Hausnummer, 0 AS Anzahl, 0 AS Flaechen
    FROM Haus h
    WHERE (h.Strasse, h.Hausnummer) NOT IN (
        SELECT Strasse, Hausnummer
        FROM view1);

SELECT * FROM ( (SELECT * FROM view1) UNION
                (SELECT * FROM view2));

DROP VIEW view1;
DROP VIEW view2;

— 2.

BEGIN
    DBMS_OUTPUT.PUT_LINE('2. Eigentuerer, der die meisten Wohnungen besitzt');
END;
/

— diese View gibt die Anzahl der Wohnungen aus, welche eine bestimmte Person besitzt,
— wenn diese Person mindestens eine Wohnung besitzt
CREATE VIEW MaxWohnungenProEigentuerer AS

```

```

SELECT MAX(COUNT(*)) AS Anzahl
FROM Person p, gehoert g
WHERE p.SVNr=g.Eigentuemer
GROUP BY p.SVNr, p.Name
HAVING COUNT(*)>=1;

SELECT p.SVNr, p.Name, COUNT(*)
FROM Person p, gehoert g
WHERE p.SVNr=g.Eigentuemer
GROUP BY p.SVNr, p.Name
HAVING COUNT(*)=(SELECT Anzahl FROM MaxWohnungenProEigentuemer);

DROP VIEW MaxWohnungenProEigentuemer;

— 3.

BEGIN
    DBMS_OUTPUT.PUT_LINE('3. Haeuser, bei denen alle Auftraege an andere Firmen erteilt sind');
END;
/

SELECT s.Strasse, s.Hausnummer
FROM Sanierungsarbeit s, Auftrag a
WHERE s.ID=a.ID
GROUP BY s.Strasse, s.Hausnummer
HAVING COUNT(a.Firma)=COUNT(DISTINCT a.Firma);

— 4.

BEGIN
    DBMS_OUTPUT.PUT_LINE('4. Alle Verwaltungsstellen mit Budget und Haeusern mit mind. 2
    Ausschreibungen oder 2 Auftraegen');
    DBMS_OUTPUT.PUT_LINE('Interpretation: Exklusives Oder:');
END;
/

— Variante: Exklusives ODER

— liefert die Haeuser, fuer die es mindestens zwei Sanierungsausschreibungen gibt
CREATE VIEW SanierungsausschrProHaus AS
    SELECT h.Strasse,
           h.Hausnummer,
           h.Verwaltungsstelle,
           v.Jahresbudget
    FROM Haus h, Sanierungsarbeit s, Ausschreibung a, Verwaltungsstelle v
    WHERE h.Strasse=s.Strasse AND
           h.Hausnummer=s.Hausnummer AND
           s.ID=a.ID AND
           h.Verwaltungsstelle=v.ID
    GROUP BY h.Strasse, h.Hausnummer, h.Verwaltungsstelle, v.Jahresbudget
    HAVING COUNT(a.ID)>=2;

— liefert die Haeuser, fuer die es mindestens zwei Sanierungsauftraege gibt
CREATE VIEW SanierungsauftraegeProHaus AS
    SELECT h.Strasse,
           h.Hausnummer,
           h.Verwaltungsstelle,
           v.Jahresbudget
    FROM Haus h, Sanierungsarbeit s, Auftrag a, Verwaltungsstelle v
    WHERE h.Strasse=s.Strasse AND
           h.Hausnummer=s.Hausnummer AND
           s.ID=a.ID AND
           h.Verwaltungsstelle=v.ID
    GROUP BY h.Strasse, h.Hausnummer, h.Verwaltungsstelle, v.Jahresbudget
    HAVING COUNT(a.ID)>=2;

SELECT Verwaltungsstelle, Jahresbudget, COUNT(*)
FROM ((SELECT Verwaltungsstelle, Jahresbudget, Strasse, Hausnummer
FROM SanierungsausschrProHaus
WHERE (Verwaltungsstelle, Strasse, Hausnummer) NOT IN ( SELECT Verwaltungsstelle, Strasse,
Hausnummer
```

```

FROM SanierungsauftraegeProHaus))
UNION
(SELECT Verwaltungsstelle , Jahresbudget , Strasse , Hausnummer
FROM SanierungsauftraegeProHaus
WHERE (Verwaltungsstelle , Strasse , Hausnummer) NOT IN ( SELECT Verwaltungsstelle , Strasse ,
Hausnummer
FROM SanierungsausschrProHaus)))
GROUP BY Verwaltungsstelle , Jahresbudget;

— Variante: Inklusives ODER

BEGIN
DBMS_OUTPUT.PUT_LINE(' Interpretation: _Inklusives_Oder: ');
END;
/

CREATE VIEW view1 AS
SELECT Verwaltungsstelle , Jahresbudget , Strasse , Hausnummer
FROM (( SELECT Verwaltungsstelle , Jahresbudget , Strasse , Hausnummer
FROM SanierungsausschrProHaus) UNION
( SELECT Verwaltungsstelle , Jahresbudget , Strasse , Hausnummer
FROM SanierungsauftraegeProHaus))
GROUP BY Verwaltungsstelle , Jahresbudget , Strasse , Hausnummer;

SELECT Verwaltungsstelle , Jahresbudget , COUNT(*)
FROM view1
GROUP BY Verwaltungsstelle , Jahresbudget;

DROP VIEW view1;
DROP VIEW SanierungsausschrProHaus;
DROP VIEW SanierungsauftraegeProHaus;

— 5.

BEGIN
DBMS_OUTPUT.PUT_LINE(' 5. _Firma_mit_den_wenigsten_Kostenvoranschlaegen ');
END;
/

SELECT f.Name, f.Firmensitz
FROM Firma f, Kostenvoranschlag k
WHERE f.Name=k.Firma AND f.Name IN (SELECT Name FROM Auftrag)
GROUP BY f.Name, f.Firmensitz
HAVING COUNT(k.Summe)<=ALL(
SELECT COUNT(k.Summe)
FROM Firma f, Kostenvoranschlag k
WHERE f.Name=k.Firma
GROUP BY f.Name, f.Firmensitz);

— 6.

BEGIN
DBMS_OUTPUT.PUT_LINE(' 6. _Anzahl_der_erteilten_Auftraege_fuer_alle_Paare_von_
Verwaltungsstellen_und_Firmen ');
END;
/

— liefert die Paare von Verwaltungsstellen und Firmen, fuer die es Auftraege gibt
CREATE VIEW View1 AS
SELECT v.ID, f.Name AS Firma, COUNT(*) AS Anzahl
FROM
Verwaltungsstelle v,
Haus h,
Sanierungsarbeit s,
Auftrag a,
Firma f
WHERE
v.ID=h.Verwaltungsstelle AND
h.Hausnummer=s.Hausnummer AND
h.Strasse=s.Strasse AND
s.ID=a.ID AND
a.Firma=f.Name
GROUP BY v.ID, f.Name;

— liefert alle anderen Paare

```

```

CREATE VIEW View2 AS
    SELECT v.ID, f.Name, 0 AS Anzahl
    FROM Verwaltungsstelle v, Firma f
    WHERE (v.ID, f.Name) NOT IN ( SELECT ID, Firma
                                FROM View1);

SELECT *
FROM ((SELECT *
      FROM View1) UNION
      (SELECT *
      FROM View2))
ORDER BY ID ASC, Firma ASC;

DROP VIEW View2;
DROP VIEW View1;

-- 7.

BEGIN
    DBMS_OUTPUT.PUT_LINE('7. Berechnen der Betriebskosten pro Person und Haus');
END;
/

-- liefert die Betriebskosten, welche eine Person an ein Haus zahlen muss
CREATE VIEW BetriebskostenDaten AS
    SELECT SVNr, Hausnummer, Strasse, SUM(Kosten) AS Kosten
    FROM ((SELECT p.SVNr, /* Betriebskosten durch Mietvertrag */
                h.Hausnummer,
                h.Strasse,
                w.Flaeche*s.Preis AS Kosten
            FROM Person p,
                mietet m,
                Wohnung w,
                Haus h,
                Sanierungskategorie s
            WHERE m.Mieter=p.SVNr AND
                  w.Hausnummer=m.Hausnummer AND
                  w.Strasse=m.Strasse AND
                  w.Tuernummer=m.Tuernummer AND
                  w.Strasse=h.Strasse AND
                  w.Hausnummer=h.Hausnummer AND
                  h.Bezeichnung=s.Bezeichnung) UNION
          (SELECT p.SVNr, /* Betriebskosten durch Eigentumerschaft */
                h.Hausnummer,
                h.Strasse,
                w.Flaeche*s.Preis AS Kosten
            FROM Person p,
                gehoert g,
                Wohnung w,
                Haus h,
                Sanierungskategorie s
            WHERE g.Eigentuemer=p.SVNr AND
                  w.Hausnummer=g.Hausnummer AND
                  w.Strasse=g.Strasse AND
                  w.Tuernummer=g.Tuernummer AND
                  w.Strasse=h.Strasse AND
                  w.Hausnummer=h.Hausnummer AND
                  h.Bezeichnung=s.Bezeichnung))
    GROUP BY SVNr, Hausnummer, Strasse;

-- berechnet die Betriebskosten, welche alle Personen an ein bestimmtes Haus zu zahlen
-- haben, und speichert diese in der Datenbank
CREATE PROCEDURE BetriebskostenFuerHaus (var_strasse VARCHAR2, var_hausnummer INTEGER) IS
    var_svnr INTEGER; /* Zwischenspeicherung der Person */
    var_kosten FLOAT; /* Betriebskosten */
    CURSOR c1 IS /* noch nicht gespeicherte Betriebskosten (neu einzutragen) */
        SELECT SVNr, Kosten
        FROM BetriebskostenDaten
        WHERE Strasse=var_strasse AND
              Hausnummer=var_hausnummer AND
              (SVNr, Hausnummer, Strasse) NOT IN ( SELECT SVNr, Hausnummer, Strasse
                                                  FROM Betriebskosten);

```

```

        CURSOR c2 IS      /* bereits eingetragene Betriebskosten (aktualisieren) */
        SELECT SVNr, Kosten
        FROM BetriebskostenDaten
        WHERE Strasse=var_strasse AND
              Hausnummer=var_hausnummer AND
              (SVNr, Hausnummer, Strasse) IN (SELECT SVNr, Hausnummer, Strasse
                                              FROM Betriebskosten);
BEGIN
    OPEN c1;
    LOOP      /* neue Betriebskosten abrufen und speichern */
        FETCH c1 INTO var_svnr, var_kosten;
        EXIT WHEN c1%NOTFOUND;
        INSERT INTO Betriebskosten VALUES (var_strasse, var_hausnummer, var_svnr, var_kosten
        );
    END LOOP;
    CLOSE c1;

    OPEN c2;
    LOOP      /* bereits eingetragene Betriebskosten aktualisieren */
        FETCH c2 INTO var_svnr, var_kosten;
        EXIT WHEN c2%NOTFOUND;
        UPDATE Betriebskosten SET Summe=var_kosten WHERE Person=var_svnr AND Hausnummer=
            var_hausnummer AND Strasse=var_strasse;
    END LOOP;
    CLOSE c2;

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
END;
/

-- berechnet fuer alle Haeuser fuer alle Personen die Betriebskosten
-- und speichert sie in der Datenbank
DECLARE
    CURSOR c1 IS
        SELECT Strasse, Hausnummer
        FROM Haus;
    var_strasse VARCHAR2(100);
    var_hausnummer INTEGER;
BEGIN
    OPEN c1;
    LOOP
        FETCH c1 INTO var_strasse, var_hausnummer;
        EXIT WHEN c1%NOTFOUND;
        BetriebskostenFuerHaus(var_strasse, var_hausnummer);
    END LOOP;
    CLOSE c1;

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
END;
/

DROP PROCEDURE BetriebskostenFuerHaus;
DROP VIEW BetriebskostenDaten;

-- 8.
BEGIN
    DBMS_OUTPUT.PUT_LINE('8. Berechnen des Sofortkaufpreises einer Wohnung mit Kaufoption');
END;
/

-- liefert fuer eine Wohnung mit Kaufoption den Sofortkaufpreis
-- Parameter: Schluessel einer Wohnung
-- Rueckgabewert: Kaufpreis, wenn Kaufoption vorhanden, sonst -1
CREATE FUNCTION SofortkaufoptionPreis (var_strasse VARCHAR2, var_hausnummer INTEGER, var_tuernummer
    INTEGER) RETURN INTEGER IS

```

```

var_preis FLOAT;          /* gespeicherter Preis */
var_laufzeit INTEGER;     /* Laufzeit der Option */
var_miete FLOAT;          /* monatliche Miete */
var_vertragsdatum DATE;   /* Abschlussdatum des Mietvertrags */
var_kaufpreis INTEGER;    /* berechneter Kaufpreis */
var_monate INTEGER;       /* Anzahl der vom Preis abgezogenen Monatsmieten */
var_abfrage INTEGER;      /* Statusvariable */

BEGIN
    var_abfrage:=1; /* Abfrage 1 wird durchgefuehrt */
    SELECT Preis, Laufzeit INTO var_preis, var_laufzeit FROM Kaufoption WHERE Strasse=
        var_strasse AND Hausnummer=var_hausnummer AND Tuernummer=var_tuernummer;
    var_abfrage:=2; /* Abfrage 2 */
    SELECT Miete, Vertragsdatum INTO var_miete, var_vertragsdatum FROM mietet WHERE Strasse=
        var_strasse AND Hausnummer=var_hausnummer AND Tuernummer=var_tuernummer;

    var_monate:=(SYSDATE-var_vertragsdatum)/31;
    IF var_monate>(var_laufzeit*12) THEN /* hoechstens Mieten fuer die Laufzeit abziehen */
        var_monate:=(var_laufzeit*12);
    END IF;
    var_kaufpreis:=var_preis-var_miete*var_monate;
    IF var_kaufpreis<0 THEN
        RETURN 0; /* Kaufpreis bereits durch Miete abbezahlt */
    ELSE
        RETURN var_kaufpreis;
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        IF var_abfrage=1 THEN
            DBMS_OUTPUT.PUT_LINE('Diese Wohnung besitzt keine Kaufoption');
            RETURN -1;
        ELSE
            RETURN var_preis; /* Wohnung ist nicht vermietet */
        END IF;
END;
/

-- berechnet fuer alle Wohnungen den Sofortkaufpreis
DECLARE
    CURSOR c1 IS
        SELECT Strasse, Hausnummer, Tuernummer
        FROM Wohnung;
    var_strasse VARCHAR2(100); /* Zwischenspeicherung der Wohnung */
    var_hausnummer VARCHAR2(4);
    var_tuernummer VARCHAR2(4);
    var_kaufpreis FLOAT; /* berechneter Kaufpreis */
BEGIN
    OPEN c1;
    LOOP
        FETCH c1 INTO var_strasse, var_hausnummer, var_tuernummer;
        EXIT WHEN c1%NOTFOUND;
        DBMS_OUTPUT.PUT('Wohnung: ');
        DBMS_OUTPUT.PUT(var_strasse);
        DBMS_OUTPUT.PUT(' ');
        DBMS_OUTPUT.PUT(var_hausnummer);
        DBMS_OUTPUT.PUT(' ');
        DBMS_OUTPUT.PUT(var_tuernummer);
        DBMS_OUTPUT.PUT(' ');
        DBMS_OUTPUT.PUT('Kaufpreis: ');
        var_kaufpreis:=SofortkaufoptionPreis(var_strasse, var_hausnummer, var_tuernummer);
        IF var_kaufpreis>=0 THEN
            DBMS_OUTPUT.PUT(var_kaufpreis);
            DBMS_OUTPUT.PUT_LINE(' EUR');
        END IF;
    END LOOP;
    CLOSE c1;
EXCEPTION
    WHEN OTHERS THEN
        RETURN;
END;
/

DROP FUNCTION SofortkaufoptionPreis;

-- 9.

```

```

BEGIN
    DBMS_OUTPUT.PUT_LINE('9. Trigger zur Vergabe weiterer Aufträge');
END;
/

-- liefert die kleinste Sanierungs-ID grösser oder gleich 1, welche noch nicht vergeben ist
-- liefert -1 im Fehlerfall
CREATE FUNCTION GetSanierungsID RETURN INTEGER IS
    CURSOR c1 IS /* grösste ID abfragen */
        SELECT s.ID
        FROM Sanierungsarbeit s
        ORDER BY s.ID ASC;
    this_id INTEGER;
    last_id INTEGER;
BEGIN
    last_id:=0;
    OPEN c1;
    LOOP
        FETCH c1 INTO this_id;
        IF this_id-last_id>1 OR c1%NOTFOUND THEN
            CLOSE c1;
            RETURN last_id+1;
        END IF;
        last_id:=this_id;
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        RETURN -1;
END;
/

-- vergibt eine ausgeschriebene Sanierungsarbeit an die bestbietende Firma
CREATE PROCEDURE AuftragVergeben(AusschreibungsID INTEGER) IS
    CURSOR c1 IS /* günstigstes Angebot */
        SELECT Firma, Summe
        FROM Kostenvoranschlag
        WHERE Ausschreibung=AusschreibungsID AND
              Summe<=ALL(
                SELECT Summe
                FROM Kostenvoranschlag
                WHERE Ausschreibung=AusschreibungsID);
    var_firma VARCHAR2(100);
    var_summe VARCHAR2(100);
    var_sanierungs_id INTEGER;
    var_verwaltungsstelle INTEGER;
    var_budget FLOAT;
    var_hausnummer INTEGER;
    var_strasse VARCHAR2(100);
    var_beschreibung VARCHAR2(255);
    /* Exceptions */
    budget_zu_gering EXCEPTION;
    kein_kostenvoranschlag EXCEPTION;
BEGIN
    OPEN c1;
    FETCH c1 INTO var_firma, var_summe;
    IF c1%NOTFOUND THEN
        CLOSE c1;
        RAISE kein_kostenvoranschlag;
    END IF;
    CLOSE c1;

    SELECT v.ID, v.Jahresbudget, s.Beschreibung, h.Hausnummer, h.Strasse
    INTO var_verwaltungsstelle, var_budget, var_beschreibung, var_hausnummer,
        var_strasse
    FROM Sanierungsarbeit s, Haus h, Verwaltungsstelle v
    WHERE s.ID=AusschreibungsID AND
          s.Strasse=h.Strasse AND
          s.Hausnummer=h.Hausnummer AND
          h.Verwaltungsstelle=v.ID;

    IF var_budget<var_summe THEN
        RAISE budget_zu_gering;
    END IF;

```

```

var_sanierungs_id:=GetSanierungsID;
var_budget:=var_budget-var_summe;

/* Aenderungen speichern */
INSERT INTO Sanierungsarbeit VALUES (var_sanierungs_ID, var_beschreibung, var_strasse,
var_hausnummer);
INSERT INTO Auftrag VALUES (var_sanierungs_ID, 'in_Vorbereitung', var_firma);
UPDATE Verwaltungsstelle SET Jahresbudget=var_budget WHERE ID=var_verwaltungsstelle;

DBMS_OUTPUT.PUT('Auftrag_wird_um_');
DBMS_OUTPUT.PUT(var_summe);
DBMS_OUTPUT.PUT('EUR_vergeben_an_');
DBMS_OUTPUT.PUT_LINE(var_firma);
DBMS_OUTPUT.PUT('Verbleibendes_Budget_der_zustaendigen_Verwaltungsstelle_');
DBMS_OUTPUT.PUT(var_verwaltungsstelle);
DBMS_OUTPUT.PUT(':_');
DBMS_OUTPUT.PUT_LINE(var_budget);
EXCEPTION
  WHEN budget_zu_gering THEN
    DBMS_OUTPUT.PUT_LINE('Budget_zu_gering!');
  WHEN kein_kostenvoranschlag THEN
    DBMS_OUTPUT.PUT_LINE('Fuer_die_beendete_Ausschreibung_existiert_kein_
    Kostenvoranschlag. ');
  WHEN OTHERS THEN
    RETURN;
END;
/

-- Trigger, der bei Ende einer Ausschreibung ausgeloeset wird
CREATE TRIGGER AusschreibungTrigger
AFTER UPDATE ON Ausschreibung
REFERENCING NEW AS newRecord OLD AS oldRecord
FOR EACH ROW WHEN (newRecord.beendet=1 AND oldRecord.beendet=0)
BEGIN
  AuftragVergeben(:newRecord.ID);
END AusschreibungTrigger;
/

-- Test fuer den Trigger
UPDATE Ausschreibung SET beendet=1 WHERE ID=3;

SPOOL OFF

```

5.4 Löschen der Datenbankstrukturen

```

/**
 * Datenbanksysteme, WS 2005, Beispiel 2
 * Paul Staroch, 0425426
 * DROP.SQL - Loeschen aller erstellten Datenbankstrukturen
 */

-- Aufgabe 9
DROP TRIGGER AusschreibungTrigger;
DROP PROCEDURE AuftragVergeben;
DROP FUNCTION GetSanierungsID;

-- Indizes der Cluster
DROP INDEX HaeuserIndex;
DROP INDEX WohnungenIndex;
DROP INDEX ArbeitenIndex;
DROP INDEX PersonenIndex;

-- Indizes der einzelnen Relationen
DROP INDEX BetriebskostenIdx;
DROP INDEX mietetIdx;
DROP INDEX gehoertIdx;
DROP INDEX KostenvoranschlagIdx;
DROP INDEX WohnungIdx;
DROP INDEX AuftragIdx;

-- Relationen

```

```

DROP TABLE Betriebskosten;
DROP TABLE mietet;
DROP TABLE gehoert;
DROP TABLE Kostenvoranschlag;
DROP TABLE Eigentuemer;
DROP TABLE Mieter;
DROP TABLE Person;
DROP TABLE Kaufoption;
DROP TABLE Wohnung;
DROP TABLE Auftrag;
DROP TABLE Firma;
DROP TABLE Ausschreibung;
DROP TABLE Sanierungsarbeit;
DROP TABLE Haus;
DROP TABLE Sanierungskategorie;
DROP TABLE Mitarbeiter;
DROP TABLE Verwaltungsstelle;

```

```

— Cluster

```

```

DROP CLUSTER Personen;
DROP CLUSTER Arbeiten;
DROP CLUSTER Wohnungen;
DROP CLUSTER Haeuser;

```

5.5 Ergebnis

1. Anzahl der vermieteten Wohnungen und Summe derer Wohnflaechen alle Haeuser

PL/SQL procedure successfully completed.

View created.

View created.

STRASSE

HAUSNUMMER	ANZAHL	FLAECHE
------------	--------	---------

Floragasse

20	1	31
----	---	----

Gusshausstrasse

31	0	0
----	---	---

Margaretenstrasse

7	1	88
---	---	----

Operngasse

18	0	0
----	---	---

Paulanergasse

2	0	0
---	---	---

Schleifmuehl-gasse

8	1	102
---	---	-----

Taubstummengasse

4	1	67
---	---	----

Waaggasse

11	2	115
----	---	-----

8 rows selected.

View dropped.

View dropped.

2. Eigentuerer, der die meisten Wohnungen besitzt

PL/SQL procedure successfully completed.

View created.

SVNR NAME	COUNT(*)
7855110875 Anna Pichler	2

View dropped.

3. Haeuser, bei denen alle Auftraege an andere Firmen erteilt sind

PL/SQL procedure successfully completed.

STRASSE

HAUSNUMMER
Floragasse
20
Taubstummengasse
4

4. Alle Verwaltungsstellen mit Budget und Haeusern mit mind. 2 Ausschreibungen oder 2 Auftraegen
Interpretation: Exklusives Oder:

PL/SQL procedure successfully completed.

View created.

View created.

VERWALTUNGSSTELLE	JAHRESBUDGET	COUNT(*)
1	100000	1
4	20000	1

Interpretation: Inklusives Oder:

PL/SQL procedure successfully completed.

View created.

VERWALTUNGSSTELLE	JAHRESBUDGET	COUNT(*)
1	100000	1
3	172000	1
4	20000	1

View dropped.

View dropped.

View dropped.

5. Firma mit den wenigsten Kostenvoranschlaegen

PL/SQL procedure successfully completed.

NAME

FIRMENSITZ

Fassadenerneuerung Gruber

Innsbruck

Hausrenovierung Bacher

Linz

6. Anzahl der erteilten Auftraege fuer alle Paare von Verwaltungsstellen und Firmen

PL/SQL procedure successfully completed.

View created.

View created.

ID FIRMA

ANZAHL

1	Fassadenerneuerung Bauer	1
1	Fassadenerneuerung Gruber	0
1	Hausrenovierung Bacher	0
1	Hausrenovierung Mueller	1
1	Installateur Maier	1
1	Installateur Schmidt	0
2	Fassadenerneuerung Bauer	0
2	Fassadenerneuerung Gruber	0
2	Hausrenovierung Bacher	0
2	Hausrenovierung Mueller	0
2	Installateur Maier	0

2	Installateur	Schmidt
0		
3	Fassadenerneuerung	Bauer
0		
3	Fassadenerneuerung	Gruber
0		
3	Hausrenovierung	Bacher
0		
3	Hausrenovierung	Mueller
2		
3	Installateur	Maier
0		
3	Installateur	Schmidt
0		
4	Fassadenerneuerung	Bauer
0		
4	Fassadenerneuerung	Gruber
0		
4	Hausrenovierung	Bacher
1		
4	Hausrenovierung	Mueller
1		
4	Installateur	Maier
0		
4	Installateur	Schmidt
0		
5	Fassadenerneuerung	Bauer
0		
5	Fassadenerneuerung	Gruber
0		
5	Hausrenovierung	Bacher
0		
5	Hausrenovierung	Mueller
0		
5	Installateur	Maier
0		
5	Installateur	Schmidt
0		
6	Fassadenerneuerung	Bauer
0		
6	Fassadenerneuerung	Gruber
0		
6	Hausrenovierung	Bacher
0		
6	Hausrenovierung	Mueller
0		
6	Installateur	Maier
0		
6	Installateur	Schmidt

0

36 rows selected.

View dropped.

View dropped.

7. Berechnen der Betriebskosten pro Person und Haus

PL/SQL procedure successfully completed.

View created.

Procedure created.

PL/SQL procedure successfully completed.

Procedure dropped.

View dropped.

8. Berechnen des Sofortkaufpreises einer Wohnung mit Kaufoption

PL/SQL procedure successfully completed.

Function created.

Wohnung: Taubstummengasse 4/11

Kaufpreis: 93712 EUR

Wohnung: Taubstummengasse 4/54

Kaufpreis: Diese Wohnung besitzt keine Kaufoption

Wohnung: Gusshausstrasse 31/13

Kaufpreis: 72500 EUR

Wohnung: Gusshausstrasse 31/16

Kaufpreis: Diese Wohnung besitzt keine Kaufoption

Wohnung: Schleifmuehlgasse 8/35

Kaufpreis: Diese Wohnung besitzt keine Kaufoption

Wohnung: Schleifmuehlgasse 8/8

Kaufpreis: Diese Wohnung besitzt keine Kaufoption

Wohnung: Operngasse 18/23

Kaufpreis: 36000 EUR

Wohnung: Operngasse 18/4

Kaufpreis: Diese Wohnung besitzt keine Kaufoption

Wohnung: Margaretenstrasse 7/11

Kaufpreis: Diese Wohnung besitzt keine Kaufoption

Wohnung: Margaretenstrasse 7/9

Kaufpreis: Diese Wohnung besitzt keine Kaufoption

Wohnung: Paulanergasse 2/32

Kaufpreis: 100000 EUR

Wohnung: Paulanergasse 2/17

Kaufpreis: Diese Wohnung besitzt keine Kaufoption

Wohnung: Waaggasse 11/49

Kaufpreis: Diese Wohnung besitzt keine Kaufoption

Wohnung: Waaggasse 11/17

Kaufpreis: 393820 EUR

Wohnung: Floragasse 20/12

Kaufpreis: 34145 EUR

Wohnung: Floragasse 20/1

Kaufpreis: Diese Wohnung besitzt keine Kaufoption

PL/SQL procedure successfully completed.

Function dropped.

9. Trigger zur Vergabe weiterer Auftraege

PL/SQL procedure successfully completed.

Function created.

Procedure created.

Trigger created.

Auftrag wird um 25000 EUR vergeben an Hausrenovierung Mueller
Verbleibendes Budget der zustandigen Verwaltungsstelle 3: 147000

1 row updated.