

Before database designing requirements must be collected and analyzed.

1 Conceptual database design

Step 1

Create local conceptual data model for every perspective

Step 1.1

Specify entity sets

Step 1.2

Specify types of relationships

Step 1.3

Specify attributes of entity sets and relationships

Step 1.4

Specify attributes domains

Step 1.5

Specify candidate keys and primary keys

Step 1.6

Consider advanced modeling methods (optional step)

Step 1.7

Verify redundancy occurrence in a model

Step 1.8

Verify realization of transactions in a local conceptual model

Step 1.9

Discuss local conceptual data model with user

2 Logical Database Design for the Relational Model

Step 2

Creation and verification of a local logical model for a whole perspective

Step 2.1

Remove properties incompatible with a relational model (optional step)

Step 2.2

Derive relations for logical data model

Step 2.3

Validate relations using normalization

Step 2.4

Validate relations against user transactions

Step 2.5

Check integrity constraints

Step 2.6

Review logical data model with user

Step 3

Merge and verify logical data models into global model

Step 3.1

Merge logical data models into global model

Step 3.2

Validate the global logical data model

Step 3.3

Check for future growth

Step 3.4

Review global, logical data model with user

3 Translate logical data model for target DBMS**Step 4**

Translate logical data model for target DBMS

Step 4.1

Design database relations

Step 4.2

Design representation of derived data

Step 4.3

Design general constraints

Step 5

Design file organizations and indexes

Step 6

Design users views

Step 7

Design security mechanisms

Step 8

Consider the introduction of controlled redundancy

Step 9

Monitor and tune the operational system

4 Examples

4.1 Clinic

Entity sets:

- Doctors,
- Specializations,
- Clinics,
- Patients,
- Appointments,
- Prescriptions,
- Medicaments,
- Addresses.

Relationships:

- Doctors - Specializations ($*$: 1 or $*$: $*$),
- Doctors - Clinics ($*$: 1 or $*$: $*$),
- Doctors - Addresses (1 : $*$ or $*$: 1 or $*$: $*$, for example registered address, home address, correspondence address),
- Clinics - Patients ($*$: $*$),
- Appointments - Doctors ($*$: 1),

- Appointments - Patients (* : 1),
- Appointments - Clinics (* : 1),
- Appointments - Prescriptions (1 : *),
- Prescriptions - Medicaments (* : *),
- Patients - Addresses (1 : * or * : 1 or * : *).

Superclass-subclass relationship:

Participation	Disjoint	Relations
Mandatory	Nondisjoint {And}	One relation (with one or more discriminators determining the type of each tuple).
Optional	Nondisjoint {And}	Two relations: one representing a superclass and the other a subclass (with one or more discriminators determining the type of each tuple).
Mandatory	Disjoint {Or}	Multiple relations: a separate relation for each superclass - subclass combination.
Optional	Disjoint {Or}	Multiple relations: a separate relation representing the superclass and separate relations for each of the subclasses.

Source: T. Connolly, C. Begg, "Systemy baz danych", Wydawnictwo RM, 2004 More information can be found in: Fowler M., "Architektura systemów zarządzania przedsiębiorstwem. Wzorce projektowe", Helion, 2005

Case 1

Mandatory - every Person in the database must be a Patient or a Doctor

Nondisjoint - Person in the database can be both a Patient and a Doctor

One relation (with one or more discriminators determining the type of each tuple).

```
CREATE TABLE Persons (
  Persons_id NUMBER(6) CONSTRAINT Persons_PK PRIMARY KEY,
  Discriminator VARCHAR2(20),
  Firstname VARCHAR2(20),
  Lastname VARCHAR2(20),
  Specializations_id NUMBER(6) NOT NULL CONSTRAINT Persons_Specializations_FK
    REFERENCES Specializations (Specializations_id),
  Insurances_id NUMBER(6) NOT NULL CONSTRAINT Persons_Insurances_FK
    REFERENCES Insurances (Insurances_id)
);
```

Listing 1: {sql/super-sub-class1.sql}

Case 2

Optional - every Person in the database can but don't have to be a Patient or a Doctor

Nondisjoint - Person in the database can be both a Patient and a Doctor

Two relations: one representing a superclass and the other a subclass (with one or more discriminators determining the type of each tuple).

```
CREATE TABLE Persons (
  Persons_id NUMBER(6) CONSTRAINT Persons_PK PRIMARY KEY,
  Firstname VARCHAR2(20),
  Lastname VARCHAR2(20),
);

CREATE TABLE Person_Details (
  Person_Details_id NUMBER(6) CONSTRAINT Persons_PK PRIMARY KEY,
  Persons_id NUMBER(6) NOT NULL CONSTRAINT Person_Details_Persons_FK
    REFERENCES Persons (Persons_id),
  Discriminator VARCHAR2(20),
  Specializations_id NUMBER(6) NOT NULL CONSTRAINT Person_Details_Specializations_FK
```

```
REFERENCES Specializations (Specializations_id),
Insurances_id NUMBER(6) NOT NULL CONSTRAINT Person_Details_Insurances_FK
REFERENCES Insurances (Insurances_id)
```

Listing 2: {sql/super-sub-class2.sql}

Case 3

Mandatory - every Person in the database must be a Patient or a Doctor

Disjoint - Person in the database cannot be both a Patient and a Doctor

Multiple relations: a separate relation for each superclass - subclass combination.

```
CREATE TABLE Doctors (
  Doctors_id NUMBER(6) CONSTRAINT Doctors_PK PRIMARY KEY,
  Firstname VARCHAR2(20),
  Lastname VARCHAR2(20),
  Specializations_id NUMBER(6) NOT NULL CONSTRAINT Doctors_Specializations_FK
  REFERENCES Specializations (Specializations_id)
);

CREATE TABLE Patients (
  Patients_id NUMBER(6) CONSTRAINT Patients_PK PRIMARY KEY,
  Firstname VARCHAR2(20),
  Lastname VARCHAR2(20),
  Insurances_id NUMBER(6) NOT NULL CONSTRAINT Patients_Insurances_FK
  REFERENCES Insurances (Insurances_id)
);
```

Listing 3: {sql/super-sub-class3.sql}

Case 4

Optional - every Person in the database can but don't have to be a Patient or a Doctor

Disjoint - Person in the database cannot be both a Patient and a Doctor

Multiple relations: Multiple relations: a separate relation representing the superclass and separate relations for each of the subclasses.

```
CREATE TABLE Persons (
  Persons_id NUMBER(6) CONSTRAINT Persons_PK PRIMARY KEY,
  Firstname VARCHAR2(20),
  Lastname VARCHAR2(20),
);

CREATE TABLE Doctors (
  Doctors_id NUMBER(6) CONSTRAINT Doctors_PK PRIMARY KEY,
  Specializations_id NUMBER(6) NOT NULL CONSTRAINT Doctors_Specializations_FK
  REFERENCES Specializations (Specializations_id)
  Persons_id NUMBER(6) NOT NULL CONSTRAINT Doctors_Persons_FK
  REFERENCES Persons (Persons_id)
);

CREATE TABLE Patients (
  Patients_id NUMBER(6) CONSTRAINT Patients_PK PRIMARY KEY,
  Insurances_id NUMBER(6) NOT NULL CONSTRAINT Patients_Insurances_FK
  REFERENCES Insurances (Insurances_id),
  Persons_id NUMBER(6) NOT NULL CONSTRAINT Patients_Persons_FK
  REFERENCES Osoba (Persons_id)
);
```

Listing 4: {sql/super-sub-class4.sql}

Relationships:

- Doctors - Specializations (* : 1 or * : *),
- Doctors - Clinics (* : 1 or * : *),
- Clinics - Patients (* : *),
- Appointments - Doctors (* : 1),
- Appointments - Patients (* : 1),
- Appointments - Clinics (* : 1),
- Appointments - Prescriptions (1 : *),

- Prescriptions - Medicaments (* : *),
- Persons - Doctors (Superclass - Subclass),
- Persons - Patients (Superclass - Subclass),
- Persons - Addresses (1 : * or * : 1 or * : *).

```

CREATE TABLE addresses (
    addresses_id NUMBER NOT NULL,
    persons_persons_id NUMBER NOT NULL
);

ALTER TABLE addresses ADD CONSTRAINT addresses_pk PRIMARY KEY ( addresses_id );

CREATE TABLE appointments (
    doctors_doctors_id NUMBER NOT NULL,
    appointments_id NUMBER NOT NULL,
    patientcards_patientcards_id NUMBER NOT NULL
);

ALTER TABLE appointments ADD CONSTRAINT appointments_pk PRIMARY KEY ( appointments_id );

CREATE TABLE clinics (
    clinics_id NUMBER NOT NULL
);

ALTER TABLE clinics ADD CONSTRAINT clinics_pk PRIMARY KEY ( clinics_id );

CREATE TABLE doctors (
    persons_persons_id NUMBER NOT NULL,
    -- ERROR: Column name length exceeds maximum allowed length(30)
    specializations_specializations_id NUMBER NOT NULL,
    doctors_id NUMBER NOT NULL,
    clinics_clinics_id NUMBER NOT NULL
);

ALTER TABLE doctors ADD CONSTRAINT doctors_pk PRIMARY KEY ( doctors_id );

CREATE TABLE medicaments (
    medicaments_id NUMBER NOT NULL
);

ALTER TABLE medicaments ADD CONSTRAINT medicaments_pk PRIMARY KEY ( medicaments_id );

CREATE TABLE patientcards (
    clinics_clinics_id NUMBER NOT NULL,
    patientcards_id NUMBER NOT NULL,
    patients_patients_id NUMBER NOT NULL
);

ALTER TABLE patientcards ADD CONSTRAINT patientcards_pk PRIMARY KEY ( patientcards_id );

CREATE TABLE patients (
    persons_persons_id NUMBER NOT NULL,
    patients_id NUMBER NOT NULL
);

ALTER TABLE patients ADD CONSTRAINT patients_pk PRIMARY KEY ( patients_id );

CREATE TABLE persons (
    persons_id NUMBER NOT NULL
);

ALTER TABLE persons ADD CONSTRAINT persons_pk PRIMARY KEY ( persons_id );

CREATE TABLE prescriptions (
    prescriptions_id NUMBER NOT NULL,
    appointments_appointments_id NUMBER NOT NULL
);

ALTER TABLE prescriptions ADD CONSTRAINT prescriptions_pk PRIMARY KEY ( prescriptions_id );

CREATE TABLE relation_22 (
    prescriptions_prescriptions_id NUMBER NOT NULL,
    medicaments_medicaments_id NUMBER NOT NULL
);

ALTER TABLE relation_22 ADD CONSTRAINT relation_22_pk PRIMARY KEY ( prescriptions_prescriptions_id,
    medicaments_medicaments_id );

CREATE TABLE specializations (
    specializations_id NUMBER NOT NULL
);

```

```

ALTER TABLE specializations ADD CONSTRAINT specializations_pk PRIMARY KEY ( specializations_id );

ALTER TABLE addresses
  ADD CONSTRAINT addresses_persons_fk FOREIGN KEY ( persons_persons_id )
    REFERENCES persons ( persons_id );

ALTER TABLE appointments
  ADD CONSTRAINT appointments_doctors_fk FOREIGN KEY ( doctors_doctors_id )
    REFERENCES doctors ( doctors_id );

ALTER TABLE appointments
  ADD CONSTRAINT appointments_patientcards_fk FOREIGN KEY ( patientcards_patientcards_id )
    REFERENCES patientcards ( patientcards_id );

ALTER TABLE doctors
  ADD CONSTRAINT doctors_clinics_fk FOREIGN KEY ( clinics_clinics_id )
    REFERENCES clinics ( clinics_id );

ALTER TABLE doctors
  ADD CONSTRAINT doctors_persons_fk FOREIGN KEY ( persons_persons_id )
    REFERENCES persons ( persons_id );

ALTER TABLE doctors
  ADD CONSTRAINT doctors_specializations_fk FOREIGN KEY ( specializations_specializations_id )
    REFERENCES specializations ( specializations_id );

ALTER TABLE patientcards
  ADD CONSTRAINT patientcards_clinics_fk FOREIGN KEY ( clinics_clinics_id )
    REFERENCES clinics ( clinics_id );

ALTER TABLE patientcards
  ADD CONSTRAINT patientcards_patients_fk FOREIGN KEY ( patients_patients_id )
    REFERENCES patients ( patients_id );

ALTER TABLE patients
  ADD CONSTRAINT patients_persons_fk FOREIGN KEY ( persons_persons_id )
    REFERENCES persons ( persons_id );

ALTER TABLE prescriptions
  ADD CONSTRAINT prescriptions_appointments_fk FOREIGN KEY ( appointments_appointments_id )
    REFERENCES appointments ( appointments_id );

ALTER TABLE relation_22
  ADD CONSTRAINT relation_22_medicaments_fk FOREIGN KEY ( medicaments_medicaments_id )
    REFERENCES medicaments ( medicaments_id );

ALTER TABLE relation_22
  ADD CONSTRAINT relation_22_prescriptions_fk FOREIGN KEY ( prescriptions_prescriptions_id )
    REFERENCES prescriptions ( prescriptions_id );

```

Listing 5: {create-table-1.sql}

- Are these instructions easy to understand?
- Where are primary keys?
- Where are foreign keys?
- What is the order in which the tables are created?

```

CREATE TABLE persons (
  persons_id NUMBER NOT NULL,
  CONSTRAINT persons_pk PRIMARY KEY ( persons_id )
);

CREATE TABLE addresses (
  addresses_id NUMBER NOT NULL,
  persons_persons_id NUMBER NOT NULL,
  CONSTRAINT addresses_pk PRIMARY KEY ( addresses_id ),
  CONSTRAINT addresses_persons_fk FOREIGN KEY ( persons_persons_id )
    REFERENCES persons ( persons_id )
);

CREATE TABLE patients (
  persons_persons_id NUMBER NOT NULL,
  patients_id NUMBER NOT NULL,
  CONSTRAINT patients_pk PRIMARY KEY ( patients_id ),
  ADD CONSTRAINT patients_persons_fk FOREIGN KEY ( persons_persons_id )
    REFERENCES persons ( persons_id )
);

CREATE TABLE clinics (
  clinics_id NUMBER NOT NULL,
  CONSTRAINT clinics_pk PRIMARY KEY ( clinics_id )
);

```

```

);

CREATE TABLE patientcards (
    clinics_clinics_id NUMBER NOT NULL,
    patientcards_id NUMBER NOT NULL,
    patients_patients_id NUMBER NOT NULL,
    CONSTRAINT patientcards_pk PRIMARY KEY ( patientcards_id ),
    CONSTRAINT patientcards_clinics_fk FOREIGN KEY ( clinics_clinics_id )
        REFERENCES clinics ( clinics_id ),
    CONSTRAINT patientcards_patients_fk FOREIGN KEY ( patients_patients_id )
        REFERENCES patients ( patients_id )
);

CREATE TABLE specializations (
    specializations_id NUMBER NOT NULL,
    CONSTRAINT specializations_pk PRIMARY KEY ( specializations_id )
);

CREATE TABLE doctors (
    persons_persons_id NUMBER NOT NULL,
    -- ERROR: Column name length exceeds maximum allowed length(30)
    specializations_specializations_id NUMBER NOT NULL,
    doctors_id NUMBER NOT NULL,
    clinics_clinics_id NUMBER NOT NULL,
    CONSTRAINT doctors_pk PRIMARY KEY ( doctors_id ),
    CONSTRAINT doctors_clinics_fk FOREIGN KEY ( clinics_clinics_id )
        REFERENCES clinics ( clinics_id ),
    CONSTRAINT doctors_persons_fk FOREIGN KEY ( persons_persons_id )
        REFERENCES persons ( persons_id ),
    CONSTRAINT doctors_specializations_fk FOREIGN KEY ( specializations_specializations_id )
        REFERENCES specializations ( specializations_id )
);

CREATE TABLE appointments (
    doctors_doctors_id NUMBER NOT NULL,
    appointments_id NUMBER NOT NULL,
    patientcards_patientcards_id NUMBER NOT NULL,
    CONSTRAINT appointments_pk PRIMARY KEY ( appointments_id ),
    CONSTRAINT appointments_doctors_fk FOREIGN KEY ( doctors_doctors_id )
        REFERENCES doctors ( doctors_id ),
    CONSTRAINT appointments_patientcards_fk FOREIGN KEY ( patientcards_patientcards_id )
        REFERENCES patientcards ( patientcards_id )
);

CREATE TABLE prescriptions (
    prescriptions_id NUMBER NOT NULL,
    appointments_appointments_id NUMBER NOT NULL,
    CONSTRAINT prescriptions_pk PRIMARY KEY ( prescriptions_id ),
    CONSTRAINT prescriptions_appointments_fk FOREIGN KEY ( appointments_appointments_id )
        REFERENCES appointments ( appointments_id )
);

CREATE TABLE medicaments (
    medicaments_id NUMBER NOT NULL,
    CONSTRAINT medicaments_pk PRIMARY KEY ( medicaments_id )
);

CREATE TABLE prescriptionitems (
    prescriptions_prescriptions_id NUMBER NOT NULL,
    medicaments_medicaments_id NUMBER NOT NULL,
    CONSTRAINT prescriptionitems PRIMARY KEY (
        prescriptions_prescriptions_id, medicaments_medicaments_id ),
    CONSTRAINT prescriptionitems_medicaments_fk FOREIGN KEY ( medicaments_medicaments_id )
        REFERENCES medicaments ( medicaments_id ),
    CONSTRAINT prescriptionitems_prescriptions_fk FOREIGN KEY ( prescriptions_prescriptions_id )
        REFERENCES prescriptions ( prescriptions_id )
);

```

Listing 6: {create-table-2.sql}

References

- [1] Jeffrey D. Ullman, Jennifer Widom, "Podstawowy wykład z systemów baz danych", WNT, 2001
- [2] Jeffrey D. Ullman, Jennifer Widom, "Systemy baz danych, Pełny wykład", WNT, 2006
- [3] Thomas Connolly, Carolyn Begg, "Systemy baz danych", Wydawnictwo RM, 2004
- [4] Kevin Loney, Bob Bryla, Oracle Database 10g, Podręcznik administratora baz danych. Helion 2008
- [5] Scott Urman, Ron Hardman, Michael McLaughlin, "Oracle Database 10g, Programowanie w języku PL/SQL", Helion, 2008
- [6] Price J, "Oracle Database 11g i SQL Programowanie", Helion, 2009

- [7] Ramez Elmasri, Shamkant B. Navathe, "Wprowadzeni do systemów baz danych", Helion, 2005
- [8] Sharon Allen, "Modelowanie danych", Helion, 2006
- [9] Jose A. Ramalho, Oracle 8i, Mikom, 2001
- [10] „Oracle®Database SQL Language Reference” 11g Release 1 (11.1) B28286-02 (b28286.pdf)
- [11] „Oracle®Database SQL Language Quick Reference” 11g Release 1 (11.1) B28285-02 (b28285.pdf)
- [12] Dokumentacja do bazy danych Oracle dostępna na stronie www.oracle.com

Materials for the subject are available at: <http://weaii-moodle.tu.kielce.pl/>