DATABASE DESIGN

31.03.2014
(Muscat, Oman)
OUTLINE

• Database and Database Management System Definition
• Database Design Steps
• Entity-Relationship (ER) Model
• Conceptional Data Design
• Logical Database Design
• Physical Database Design
• Normalization
• Denormalization
• DB Design Examples and Key Points
What is Database?

A **database** is an organized collection of data.

“**It is hot today**” => Not a data  
“**It is 5 C degrees**” => Data

One can insert, update or delete the data in the database directly or via a web program, etc.

Today, databases are totally in our lifes: internet shopping, banking, administrative registers etc.
Database Management System (DBMS) is a software designed to assist, managing, maintaining data.

**Main Functions of the DBMS:**

- Make new databases
- Define the concept of the database
- Store data (Different types of data available)
- Protect data
- Query data
- Encrypt data
- Controlling access rights
- Synchronize accesses
- Organization of physical data structure

An alternative to DBMS may be using text files that are less complex.
Advantages of DBMS

✓ Can deploy huge size of data
✓ Controlling redundancy
✓ Data independence from applications
✓ Reduced application development time
✓ Efficient data access, techniques for efficient query
✓ Data integrity with using referential integrity techniques (data consistency)
✓ Security, authorization for multiple users
✓ Easy data administration
✓ Backup and recovery mechanism
Disadvantages of DBMS

Size – very large
Complex
High cost
Increased hardware requirements
DATABASE DESIGN STEPS

✓ Think before doing it!

- Requirement Analysis

- Conceptional Design:
  - Relational Model (E-R Modelling)
  - Hierarchical Model
  - Network Model
  - Object Oriented Model

- Logical Design

- Physical Design
Requirement Analysis

During this phase, the below questions must be answered:

What will the system serve for?
Which requirements will this database meet?
Which data will this database store?
What will the tables of this database be?

Upon completing the answers for these questions on paper, passing to the conceptional design can be advantageous for your work.
ENTITY-RELATIONSHIP MODEL

ER Model is the concept of the Relational Databases.

ER Model describes data model using objects (Entity) and their relationships.

Entity:
- is an object that exists and can be distinguished from other objects
- has attributes

Relationship:
- Relate two or more entities
- Relationship may have attributes
ER Model

Student

- sid
- name

Course

- cid
- cname

Student

- sid
- name

enrolled
Attributes

An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.

Example:

\[
\text{instructor} = (ID, \text{name}, \text{street}, \text{city}, \text{salary})
\]
\[
\text{course} = (\text{course}\_\text{id}, \text{title}, \text{credits})
\]

**Domain** – the set of permitted values for each attribute
Attributes Types

- **Simple** and **composite, component** attributes.

- **Single-valued** and **multivalued** attributes

  Example: multivalued attribute: *phone_numbers*

- **Derived** attributes
  Can be computed from other attributes

  Example: *age*, given *date_of_birth*
Composite Attributes

**Composite attributes**

- name
  - first_name
  - middle_initial
  - last_name

**Component attributes**

- address
  - street
  - city
  - state
  - postal_code

- street_number
- street_name
- apartment_number
Entity With Composite, Multivalued, and Derived Attributes

<table>
<thead>
<tr>
<th>instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>first_name</td>
</tr>
<tr>
<td>middle_initial</td>
</tr>
<tr>
<td>last_name</td>
</tr>
<tr>
<td>address</td>
</tr>
<tr>
<td>street</td>
</tr>
<tr>
<td>street_number</td>
</tr>
<tr>
<td>street_name</td>
</tr>
<tr>
<td>apt_number</td>
</tr>
<tr>
<td>city</td>
</tr>
<tr>
<td>state</td>
</tr>
<tr>
<td>zip</td>
</tr>
<tr>
<td>{ phone_number }</td>
</tr>
<tr>
<td>date_of_birth</td>
</tr>
<tr>
<td>age ( )</td>
</tr>
</tbody>
</table>
Mapping Cardinality Constraints

- **Relationship** can be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many to many
One-to-One Relationship

• one-to-one relationship between an instructor and a student

• an instructor is associated with at most one student via advisor

• and a student is associated with at most one instructor via advisor
One-to-Many Relationship

• one-to-many relationship between an instructor and a student

• an instructor is associated with several (including 0) students via advisor

• a student is associated with at most one instructor via advisor
Many-to-Many Relationship

• An instructor is associated with several (possibly 0) students

• A student is associated with several (possibly 0) instructors via advisor relationship
Concepanional Data Design with E-R Model

*Requirement Analysis

Decide entity, attributes and relations

Entity-Relation (E-R) diagram
ER Diagram Example:

![ER Diagram Example](image-url)
Logical Database Design

- Conceptional Design is used
- ER Model is converted to Relational Database Model.
- Entity at E-R or Classes in UML --> Table
- Many to many relations --> Table
- Attributes --> Columns
- Primary keys and foreign keys are defined
Physical Database Design

• It is **Physical** implementation of the Logical Model.

• Tables, columns, primary key constraints, foreign key constraints, check constraints, unique constraints, comments are created.

• **Normalization Rules** are checked, redundancy is minimized

• Disk capacity, partition strategy, security strategy are considered.

• Performance tuning is done.
Most Common Relational Databases

- MySQL
- PostgreSQL
- Access
- Oracle
- IBM Db2
- Interbase
- Microsoft SQL Server
Normalization

• process of organizing a database into tables correctly

• Finally unproblematic tables are designed that providing Consistency, minimize redundancy

• We decide which attributes are used in a table
Normalization Forms

• (1NF) First Normal Form
• (2NF) Second Normal Form
• (3NF) Third Normal Form
• (BCNF) Boyce Codd Normal Form
• (4NF) Fourth Normal Form
• (5NF) Fifth Normal Form
First Normal Form

• Domain is **atomic** in First Normal Form if its elements are considered to be indivisible units

• UNF – Unnormalized Form Example:

<table>
<thead>
<tr>
<th>staffNo</th>
<th>job</th>
<th>dept</th>
<th>dname</th>
<th>city</th>
<th>contact number</th>
</tr>
</thead>
<tbody>
<tr>
<td>S01</td>
<td>Salesm an</td>
<td>10 sales</td>
<td>London</td>
<td>12345,767642,4982423</td>
<td></td>
</tr>
<tr>
<td>S02</td>
<td>Manage r</td>
<td>20 accounts</td>
<td>Barking</td>
<td>351632165</td>
<td></td>
</tr>
<tr>
<td>S03</td>
<td>Clerk</td>
<td>20 accounts</td>
<td>Barking</td>
<td>383131267</td>
<td></td>
</tr>
<tr>
<td>S04</td>
<td>Clerk</td>
<td>30 operations</td>
<td>Barking</td>
<td>383131267</td>
<td></td>
</tr>
</tbody>
</table>
First Normal Form Example

Sales Table:

<table>
<thead>
<tr>
<th>id</th>
<th>Name</th>
<th>Address</th>
<th>City</th>
<th>Product</th>
<th>Quantity</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ahmet Seker</td>
<td>Address1</td>
<td>Ankara</td>
<td>CD-R</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>Ahmet Seker</td>
<td>Address1</td>
<td>Ankara</td>
<td>Mouse</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Ahmet Seker</td>
<td>Address1</td>
<td>Ankara</td>
<td>CD-R</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>Ramazan Kaya</td>
<td>Address2</td>
<td>İstanbul</td>
<td>DVD-R</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>5</td>
<td>Gokhan Imam</td>
<td>Address3</td>
<td>Adana</td>
<td>CD-R</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Contains repeating data
- There are anomalies while inserting, updating and deleting
- Let’s think about avoiding repeating data ..
Second Normal Form Example

• Repeating data is prevented

Customer table:

<table>
<thead>
<tr>
<th>id</th>
<th>Name</th>
<th>Address</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ahmet Seker</td>
<td>Address1</td>
<td>Ankara</td>
</tr>
<tr>
<td>2</td>
<td>Ramazan Kaya</td>
<td>Address2</td>
<td>İstanbul</td>
</tr>
<tr>
<td>3</td>
<td>Gokhan Imam</td>
<td>Address3</td>
<td>Adana</td>
</tr>
</tbody>
</table>

Sales table:

<table>
<thead>
<tr>
<th>Customer id</th>
<th>Product</th>
<th>Quantity</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CD-R</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>Mouse</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>CD-R</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>DVD-R</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td>CD-R</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Let’s think about: What is the anomalies at these tables? How to avoid?
Anomalies at the Second Normal Form

• If adding new city is wanted, a customer should be added

• When deleting a customer, the city will also be deleted

• So?
Third Normal Form

- Tables are divided into new tables though there is no functionally dependencies
- In the example City table will be added

**City:**

<table>
<thead>
<tr>
<th>City id</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>Ankara</td>
</tr>
<tr>
<td>c2</td>
<td>İstanbul</td>
</tr>
<tr>
<td>c3</td>
<td>Adana</td>
</tr>
</tbody>
</table>

**Customer:**

<table>
<thead>
<tr>
<th>id</th>
<th>Name</th>
<th>City id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ahmet Seker</td>
<td>C1</td>
</tr>
<tr>
<td>2</td>
<td>Ramazan Kaya</td>
<td>C2</td>
</tr>
<tr>
<td>3</td>
<td>Gokhan Imam</td>
<td>C3</td>
</tr>
</tbody>
</table>
Normalization

Advantages:

✓ More efficiently
✓ More accurate data
✓ Less hard drive
✓ Fewer data integrity problems

Disadvantages:

✓ More slower
✓ More complex queries
✓ More work is needed
✓ Unless normalizing, still do its business!
Denormalization for Performance

• May want to use non-normalized schema for performance

• More tables require more joining operations while querying
What is Denormalization?

• It is a strategy that is used to increase the performance of a database infrastructure.

• Involves adding redundant data.

• Involves combining data from various tables into a single table.
Summary of Physical Database Design

Divide your information into tables regarding main subjects or entities.

Decide which columns will take place in each table
(eg. Surname and StartDate for EMPLOYEES table)

Decide the primary keys for all tables.
A primary key is used to define a record specifically
(eg. Province_code in PROVINCES tables)

Establish the table relations
Analyze each table in order to decide which columns will take place in other tables.

Detail your design
Analyze your design for errors. Create the tables and insert test records to find whether there are anomalies in your design. Make arrangements on your design if necessary.
Summary of Database Design (Cont’d)

• Apply normalization rules

• Define constraints for integrity:
  • not null
  • primary key
  • unique constraint
  • check constraints
  • Foreign keys (References another tables)

• Use indexes for performance (DB already creates for PK)
• There is **master-child** relation between university and student
• University codes of the Student should be contained in University table
• PK and Index are needed
DATABASE DESIGN EXAMPLES AND MAIN POINTS
Divide your information into tables

For example, the main entities or subjects for a Product Sales Database can be designed as below at first:

- **CUSTOMERS**
  - Name
  - City
  - Country
  - E-mail
  - Send e-mail?
  - MemberDate

- **PRODUCTS**
  - Product Name
  - Price
  - In stock?

- **MANUFACTURERS**
  - Company Name
  - Contact Person
  - Address
  - City
  - Country

- **ORDERS**
  - Order No
  - Salesperson
  - OrderDate
  - Product
  - Quantity
  - Price
  - Total Price

- These 4 entities seem good for a start
Divide your information into tables (Cont’d)

If you had designed a single table instead of 4 different tables:

Each record would contain data about both products and manufacturers.
You may have many products coming from a manufacturer.
In this case, you need to enter the name and address **multiple times** causing unnecessary disk usage. Instead, a MANUFACTURERS table related with a PRODUCTS table would provide a single record for a manufacturer.

One other anomaly would be seen while **manipulating data**.
If the address of a company changes, you need to update all the records related with that company.
Divide your information into tables (Cont’d)

If you had designed a single table instead of 4 different tables:

Another anomaly: Assume a manufacturer has only one product. If you want to delete this product, but want to keep the Manufacturer’s data, you can not achieve this goal.

Note: create tables(entities) that represent a subject and include columns only related to that subject.

For example, Manufacturer address is a concept that belongs to manufacturer, not to product. Hence, this column should be in MANUFACTURERS table.
Decide which columns will take place in each table

Assume you decided your address column to include country, province, and districts in a single column (Turkey, Ankara, Çankaya).

If you will produce reports based on province or order the reports by country, than you’d better divide this field into 3 separate columns. (Codes of these locations are preferable)

Separate Name and Surname columns if surnames are important in your future reports.

No need to include derived columns in tables (eg. Age, Total Price)
For example, the main entities or subjects for a Product Sales Database can be designed as below at first:

- **CUSTOMERS**
  - Name
  - Surname
  - City
  - Country
  - E-mail
  - Send e-mail?
  - MemberDate

- **PRODUCTS**
  - Product Name
  - Unit Price
  - # in stock
  - # in order

- **MANUFACTURERS**
  - Company Name
  - Contact Person
  - Address
  - City
  - Country

- **ORDERS**
  - Order No
  - Salesperson
  - OrderDate
  - Product
  - Quantity
  - Price

every table, you are ready to choose the Primary Key of each.
Decide Primary Keys

Primary keys are the main factors that define a record uniquely.

For example, in a PERSONNEL table, SSN is an ideal candidate for being a PK. PKs cannot be null(empty) and cannot have a repeating value in a table.

Name is a bad candidate for being a PK for a PERSONNEL, because you will most probably have records that have the same name.

The PK for a table will probably be a reference (foreign key) for another table. So, your PK’s should be unchangeable, generally.

An auto-increment number (aka. surrogate key) can also be a PK for some tables (eg. ORDERS table).

In some cases, your PK may consist of more than one column.
Our tables have Primary Keys:

- CUSTOMERS: CustomerNo, Name, Surname, City, Country, E-mail, Send e-mail?, MemberDate
- PRODUCTS: ProductNo, Product Name, Unit Price, # in stock, # in order
- MANUFACTURERS: ManufacturerNo, Company Name, Contact Person, Address, City, Country, E-mail, Phone Number
- ORDERS: OrderNo, Salesperson, OrderDate, Product, Quantity, Price
RELATE THE TABLES

On relational databases, you divide your data into subject-based tables.

Afterwards, you relate the tables in order to query more than one table at a time.
Different parts from different tables at Application
In our Product Orders database, there is MANUFACTURERS table and PRODUCTS table.

A manufacturer can produce more than one products. As a result, many rows may exist in PRODUCTS table that belong to a manufacturer.

(1- M Relationship)
1-M Relationship

A Foreign Key must be the Primary key of another table. It is the most common relationship used when creating relational databases. A row in a table in a database can be associated with one or (likely) more rows in another table.
M-M Relationship

Let’s decide the relationship between PRODUCTS and ORDERS tables. A single order may include more than one product. For every record in PRODUCTS table, more than one record in ORDERS table may exist.

On the other hand, a product may be involved in more than one order. For every record in ORDERS table, more than one record in PRODUCTS table may exist.

Let’s think what kind of problems may arise?
PREVENT DUPLICATION

For example, in the Citizen table, We chose tck_no as pk.
PREVENT DUPLICATION (Cont’d)

After making the TCK_NO column the Primary Key, if we want to insert another record with the same TCK_NO:

<table>
<thead>
<tr>
<th>TCK_NO</th>
<th>ADI</th>
<th>SOYADI</th>
<th>ANAADI</th>
<th>BABAADI</th>
</tr>
</thead>
<tbody>
<tr>
<td>19625599578</td>
<td>İLKER</td>
<td>GÜVEN</td>
<td>ŞİNASI</td>
<td>ÇİGDEN</td>
</tr>
<tr>
<td>19625599578</td>
<td>ESMER</td>
<td>GENÇ</td>
<td>CEYRAN</td>
<td>ŞEMİSTAN</td>
</tr>
<tr>
<td>34550102426</td>
<td>HATİCE</td>
<td>KAYSİZ</td>
<td>FATMA</td>
<td>MUSA</td>
</tr>
<tr>
<td>34745095984</td>
<td>RAHİME</td>
<td>TAŞKIN</td>
<td>AYŞE</td>
<td>ŞEVKET</td>
</tr>
<tr>
<td>34738982372</td>
<td>AYŞE GİI</td>
<td>TAŞKIN</td>
<td>RAHİME</td>
<td>HANAN DOĞAN</td>
</tr>
</tbody>
</table>
After making the TCK_NO column the Primary Key, if we want to insert another record with the same TCK_NO:

<table>
<thead>
<tr>
<th>TCK_NO</th>
<th>ADI</th>
<th>SOYADı</th>
<th>ANAADı</th>
<th>BABAADı</th>
</tr>
</thead>
<tbody>
<tr>
<td>* 19625599578</td>
<td>İLKER</td>
<td>GÜVEN</td>
<td>ŞINASI</td>
<td>ÇİÇİDEM</td>
</tr>
<tr>
<td>19625599578</td>
<td>ESMER</td>
<td>GENÇ</td>
<td>CEYTRAN</td>
<td>ŞEMİSTAN</td>
</tr>
<tr>
<td>34550102426</td>
<td>HATİCE</td>
<td>KAYGİSIZ</td>
<td>FATMA</td>
<td>MUSA</td>
</tr>
<tr>
<td>34745059584</td>
<td>RAHİME</td>
<td>TAŞKİN</td>
<td>AYŞE</td>
<td>ŞEVKET</td>
</tr>
<tr>
<td>34733096102</td>
<td>AYŞE GÜL</td>
<td>TAŞKİN</td>
<td>RAHİME</td>
<td>HASAN DOĞAN</td>
</tr>
<tr>
<td>34736096266</td>
<td>HAYRİYE</td>
<td>TAŞKİN</td>
<td>RAHİME</td>
<td>HASAN DOĞAN</td>
</tr>
<tr>
<td>19622599632</td>
<td>MEHMET ALİ</td>
<td>GENÇ</td>
<td>ESMER</td>
<td>SERDAR</td>
</tr>
<tr>
<td>34742086038</td>
<td>AYDIN</td>
<td>TAŞKİN</td>
<td>RAHİME</td>
<td>HASAN DOĞAN</td>
</tr>
<tr>
<td>19619599706</td>
<td>VİLDAN İSMET</td>
<td>GENÇ</td>
<td>ESMER</td>
<td>SERDAR</td>
</tr>
<tr>
<td>3473096320</td>
<td>DANIYE</td>
<td>TAŞKİN</td>
<td>HATİCE</td>
<td>FUAT</td>
</tr>
</tbody>
</table>

**TOAD Error**

ORA-00001: unique constraint (DERS.NUFUS_PK) violated
FORCE TO ENTER A VALUE

- Another consistency check mechanism is to force the user to enter a value to columns.

- For example, no records with null values in NAME and SURNAME columns should exist.

To enforce this is also possible with a database constraint.
<table>
<thead>
<tr>
<th>ID</th>
<th>Column Name</th>
<th>DataType</th>
<th>Size</th>
<th>Byte/Code</th>
<th>Precision</th>
<th>Scale</th>
<th>Not Null</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TCK_NO</td>
<td>NUMBER</td>
<td></td>
<td></td>
<td>11</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ADI</td>
<td>VARCHAR2</td>
<td>30</td>
<td>Byte</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SOYADI</td>
<td>VARCHAR2</td>
<td>30</td>
<td>Byte</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ANAADI</td>
<td>VARCHAR2</td>
<td>30</td>
<td>Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>BABAADI</td>
<td>VARCHAR2</td>
<td>30</td>
<td>Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FORCE TO ENTER A VALUE (cont’d)

<table>
<thead>
<tr>
<th>TCK_NO</th>
<th>ADI</th>
<th>SOYADI</th>
<th>ANAADI</th>
<th>BABAADI</th>
</tr>
</thead>
<tbody>
<tr>
<td>9999999</td>
<td>ESMER</td>
<td>GENÇ</td>
<td>CEYLAN</td>
<td>ŞEMİSTAN</td>
</tr>
<tr>
<td>1962559578</td>
<td>HATICE</td>
<td>KAYGİSİZ</td>
<td>FATMA</td>
<td>MUSA</td>
</tr>
<tr>
<td>34550102426</td>
<td>RAHİME</td>
<td>TAŞKIN</td>
<td>AYŞE</td>
<td>ŞEVKET</td>
</tr>
<tr>
<td>34745095984</td>
<td>AYŞE GÜL</td>
<td>TAŞKIN</td>
<td>RAHİME</td>
<td>HASAN DOĞAN</td>
</tr>
<tr>
<td>3473906102</td>
<td>HAYRİYE</td>
<td>TAŞKIN</td>
<td>RAHİME</td>
<td>HASAN DOĞAN</td>
</tr>
<tr>
<td>1962259932</td>
<td>MEHMET ALİ</td>
<td>GENÇ</td>
<td>ESMER</td>
<td>SERDAR</td>
</tr>
<tr>
<td>3474206038</td>
<td>AYDIN</td>
<td>TAŞKIN</td>
<td>RAHİME</td>
<td>HASAN DOĞAN</td>
</tr>
<tr>
<td>1961959907</td>
<td>VILDAN ISMET</td>
<td>GENÇ</td>
<td>ESMER</td>
<td>SERDAR</td>
</tr>
<tr>
<td>3473306320</td>
<td>DANIYE</td>
<td>TAŞKIN</td>
<td>HATICE</td>
<td>FUAT</td>
</tr>
<tr>
<td>2746733884</td>
<td>AHMET BUĞRA</td>
<td>TAŞKIN</td>
<td>DANIYE</td>
<td>AYDIN</td>
</tr>
<tr>
<td>1617271500</td>
<td>İZZET</td>
<td>TÜRKER</td>
<td>BÜKİRME</td>
<td>MEHMET</td>
</tr>
<tr>
<td>3151420369</td>
<td>MOHİTTİN</td>
<td>TURAN</td>
<td>ŞERİFE</td>
<td>KAZIM</td>
</tr>
<tr>
<td>3254616928</td>
<td>ŞERİFE</td>
<td>TURAN</td>
<td>MERYEM</td>
<td>IBRAHİM</td>
</tr>
<tr>
<td>3250417062</td>
<td>DÖNE</td>
<td>TURAN</td>
<td>HATICE</td>
<td>ALI İHSAN</td>
</tr>
<tr>
<td>3248017144</td>
<td>ŞERİFE</td>
<td>TURAN</td>
<td>DÖNE</td>
<td>MOHİTTİN</td>
</tr>
</tbody>
</table>

Error: ORA-01400: cannot insert NULL into ("DERS", "NÜFUS", "ADI")
RELATIONAL DATABASES

Assume we have a PERSONEL table that stores personnel data and a DEPT table that stores the departments in the corporation. These 2 tables have such records:
RELATIONAL DATABASES (Cont’d)

<table>
<thead>
<tr>
<th>PERS_NO</th>
<th>MAAS</th>
<th>AD</th>
<th>SOYAD</th>
<th>GOREV</th>
<th>DEPT_NO</th>
<th>MUDUR_NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>7512</td>
<td>3000</td>
<td>MURAT</td>
<td>BAŞKAN</td>
<td>MÜDÜR</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>8143</td>
<td>2500</td>
<td>AYŞE</td>
<td>CANDAN</td>
<td>PROGRAMCI</td>
<td>10</td>
<td>7512</td>
</tr>
<tr>
<td>7554</td>
<td>2900</td>
<td>CAN</td>
<td>DOĞAN</td>
<td>UZMAN</td>
<td>20</td>
<td>6141</td>
</tr>
<tr>
<td>6141</td>
<td>3100</td>
<td>BELGİN</td>
<td>DORUK</td>
<td>MÜDÜR</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>8112</td>
<td>1200</td>
<td>HÜLYA</td>
<td>BULUT</td>
<td>VERİ GİRİŞ OPERATÖRÜ</td>
<td>10</td>
<td>7512</td>
</tr>
<tr>
<td>6943</td>
<td>1250</td>
<td>TÜLAY</td>
<td>YIĞIT</td>
<td>VERİ GİRİŞ OPERATÖRÜ</td>
<td>20</td>
<td>6141</td>
</tr>
</tbody>
</table>
If you do not put a constraint, one day you may face with a personnel who has a department that does not exist in the DEPT table. Because, for example, data entry people may enter incorrectly while entering quickly.

A database constraint must be put for not to face with such a problem. The most important feature of the relational database is the ability to relate these tables and does not allow inconsistent data.
RELATIONAL DATABASES (Cont’d)

<table>
<thead>
<tr>
<th>PERS_NO</th>
<th>MAAS</th>
<th>AD</th>
<th>SOYAD</th>
<th>GOREV</th>
<th>DEPT_NO</th>
<th>MUDUR_NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>8159</td>
<td>2000</td>
<td>CEYDA</td>
<td>MAYDA</td>
<td>UZMAN YARDIMCISI</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>7512</td>
<td>3000</td>
<td>MURAT</td>
<td>BAŞKAN</td>
<td>MÜDÜR</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>8143</td>
<td>2500</td>
<td>AYŞE</td>
<td>CANDAN</td>
<td>PROGRAMCI</td>
<td>10</td>
<td>7512</td>
</tr>
<tr>
<td>7554</td>
<td>2900</td>
<td>CAN</td>
<td>DOĞAN</td>
<td>UZMAN</td>
<td>20</td>
<td>6141</td>
</tr>
<tr>
<td>6141</td>
<td>3100</td>
<td>BELGİN</td>
<td>DORUK</td>
<td>MÜDÜR</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>8112</td>
<td>1200</td>
<td>HÜLYA</td>
<td>BULUT</td>
<td>VERİ GİRİŞ OPERATÖRÜ</td>
<td>10</td>
<td>7512</td>
</tr>
<tr>
<td>6943</td>
<td>1250</td>
<td>TÜLAY</td>
<td>YİĞİT</td>
<td>VERİ GİRİŞ OPERATÖRÜ</td>
<td>20</td>
<td>6141</td>
</tr>
</tbody>
</table>
RELATIONAL DATABASES (Cont’d)
To prevent such an inconsistency, DEPT_NO column of PERSONEL table should be related to DEPT_NO column of DEPT table.
RELATIONAL DATABASES (Cont’d)
RELATIONAL DATABASES (Cont’d)

- After relating these 2 tables, the DEPT table is called as master table or parent table or reference table.

- PERSONEL table is called as child table.

- After this relation, no one can enter a value to PERSONEL.DEPT_NO column that does not exist in DEPT.DEP_NO column.
## RELATIONAL DATABASES (Cont’d)

### Table: PERSONEL

<table>
<thead>
<tr>
<th>PERS_NO</th>
<th>MAAS</th>
<th>AD</th>
<th>SOYAD</th>
<th>GOREV</th>
<th>DEPT_NO</th>
<th>MUDUR_NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>8159</td>
<td>2000</td>
<td>CEYDA</td>
<td>MAYDA UZMAN YARDIMCISI</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7512</td>
<td>3000</td>
<td>MURAT</td>
<td>BASKAN MUDUR</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8143</td>
<td>2500</td>
<td>AYESE</td>
<td>CANDAN PROGRAMCI</td>
<td>10</td>
<td>7512</td>
</tr>
<tr>
<td></td>
<td>7554</td>
<td>2900</td>
<td>CAN</td>
<td>DOGAN UZMAN</td>
<td>20</td>
<td>6141</td>
</tr>
<tr>
<td></td>
<td>6141</td>
<td>3100</td>
<td>BELGIN</td>
<td>DORUK MUDUR</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8112</td>
<td>1200</td>
<td>HULYA</td>
<td>BULUT VERI Giris OPERATORU</td>
<td>10</td>
<td>7512</td>
</tr>
<tr>
<td></td>
<td>6943</td>
<td>1250</td>
<td>TULAY</td>
<td>YIGIT VERI Giris OPERATORU</td>
<td>20</td>
<td>6141</td>
</tr>
</tbody>
</table>

### Error Message

```
ORA-02291: integrity constraint (DERS.PERSONEL_R01) violated - parent key not found
```

### Error handling options

- Help
- Clipboard
- Details
- OK
RELATIONAL DATABASES (Cont’d)

- But similarly, no one can also delete a record from DEPT table that has a child record in PERSONEL table.
RELATIONAL DATABASES (Cont’d)

• In a data entry application, the ideal solution for consistent data collection is to create **reference tables** and establish a PK-FK relation between the tables.

• To give an example, if you are making a population census project and the variables on the survey paper include EDUCATION LEVEL, COUNTRY CODE, GENDER the ideal design should be something like this:
RELATIONAL DATABASES (Cont’d)
RELATIONAL DATABASES (Cont’d)

- By this way, you guarantee that unknown values of education level, country code or gender can not be entered to POPULATION table.

- You can write your own constraints also in the application programming language like Java, .NET, etc.

- But it will be waste of time since any RDBMS guarantees these issues.
CONCLUSION

Database Design Steps are:

• Required Analysis
• Conceptional Data Design (ER)
• Logical Database Design (Relational)
• Physical Database Design

Before designing a database; requirements analysis should be done carefully!

Maintenance should be considered while designing

Normalization/Denormalization should be considered, evaluated.

Data integrity, consistency should be guaranteed using constraints

Performance of the system should be also evaluated
Case Study:

We are designing DB system for a shop. System includes products and sales information.

1) Please draw the ER diagram for the system
2) After ER diagram; write table names, columns, underline the primary keys and foreign keys columns
The requirements of the system are:

• Each product has product number, trademark, model, properties, unit price, guarantee period and stock quantity.

• Each product belongs to a product group, a product group may contain many products.

• The information of the customers that buy products are saved. Customers have id, name, surname, address, phone information. When a customer buy a product, invoice is arranged. More than one invoice may be arranged for a customer, but an invoice is arranged for one customer.

• There may be more products in an invoice. A product may exist more than one invoice. The price of the sale and the quantity are recorded.

• More than one payment is available for a sale. Each payment has date and paid price.

** Hint : You may Underline objects to find entities and their attributes**
Tables of the database:

• Customer (customerId, name, surname, address, phone)

• Invoice (invoiceId, invoiceDate, customerId)

• Product (productId, trademark, model, properties, unitPrice, guaranteePeriod, quantity, productGroupId)

• ProductGroup (productGroupId, name)

• Payment (invoiceId, paymentNumber, date, amount)

• Sale (saleId, invoiceId, productId, quantity, salePrice)