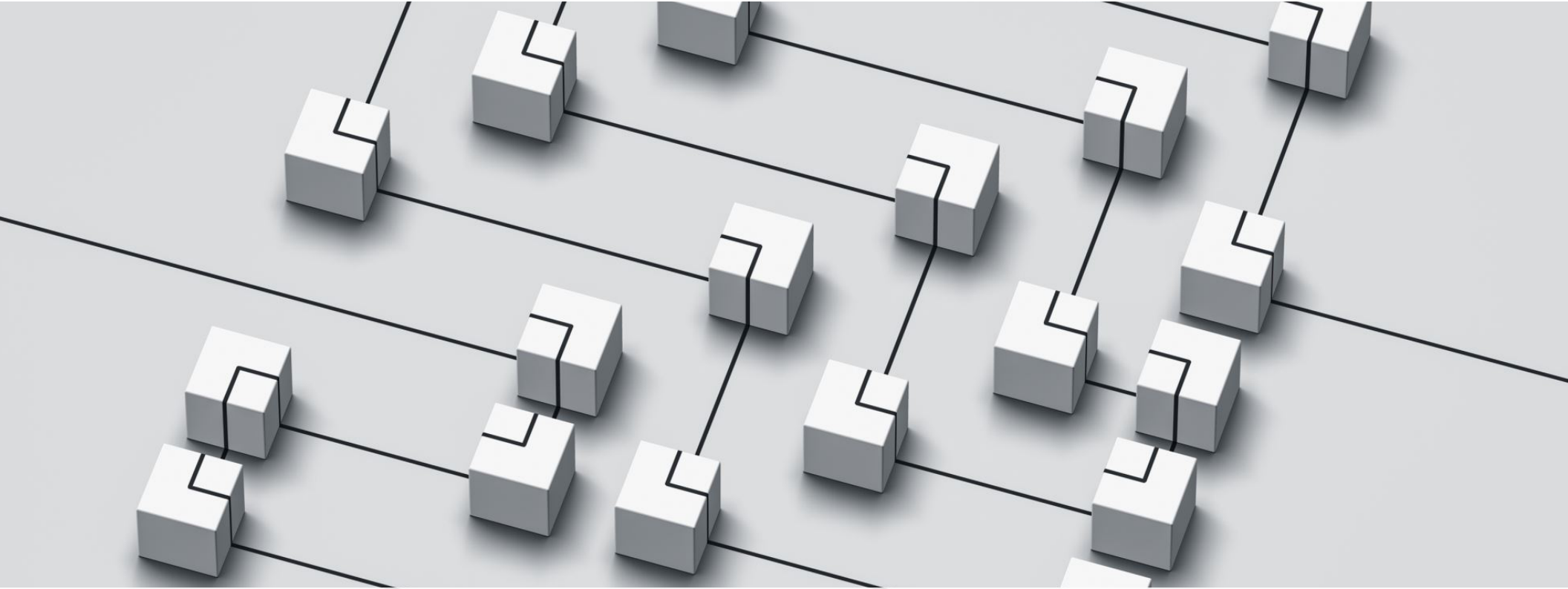


04 RELATIONAL MODEL
CS 340: INTRODUCTION TO DATABASE SYSTEMS

Adapted from: Dr. Omar Alomeir
2023

Course instructor:
Dr. Maram Alajlan.

PROJECT NEXT STEPS (SPECIFICATIONS)

Submission of team names, tech, and ideas today.

Specification for next step will be uploaded to Moodle shortly in sha’ Allah.

Read it carefully and follow all the steps.

Pay attention to the deadline.

Submission through Moodle.

LEARNING GOALS

CLO1: Define the "database", its architecture, the main concepts and characteristics of databases, as well as concepts related to the design, and implementation of a relational databases. (Knowledge & Understanding)

Chapter objectives:

- Compare logical and physical data independence.
- Define the components of relational model: tables, rows, columns, etc.
- Briefly describe DBMS languages.
- Learn about database language terms.
- Learn about relational database constraints.
- Learn the difference between database architectures.

BUILDING AN APPLICATION WITH A DBMS

Requirements modeling (conceptual, pictures)

- Decide what entities should be part of the application and how they should be linked.

Schema design and implementation (Logical)

- Decide on a set of tables, attributes.
- Define the tables in the database system.
- Populate database (insert tuples).

Write application programs using the DBMS

- Easier now that the data management is taken care of.

DATABASE DESIGN

Why do we need it?

- Agree on structure of the database before deciding on a particular implementation.

Consider issues such as:

- What entities to model
- How entities are related
- What constraints exist in the domain
- How to achieve ***good*** designs

ER diagrams can be translated to relational schema. We will see how.

We will start translating ER diagrams to SQL very soon.

LOGICAL DESIGN

- We want to store our data without worrying about files on disk.
- We want to query (retrieve) our data.
- We want a representation that is easy to understand.
- We want to easily map our ER diagrams.
- We want this to be separate from application code.

RELATIONAL MODEL

- Older models (network, hierarchical) were complex and involved a lot of application code, were difficult to query.
- Relational model introduced by Edgar Codd in 1970.
- Competitors are not nearly as popular or used today (NoSQL, XML).
- Three main properties:
  - Easy to understand and use. The main abstract element is a table.
  - Physical data independence: Modify physical schema without affecting logical schema.
  - Logical data independence: Modify conceptual schema without changing applications (views).

DBMS LANGUAGES

We will briefly look at DBMS languages:

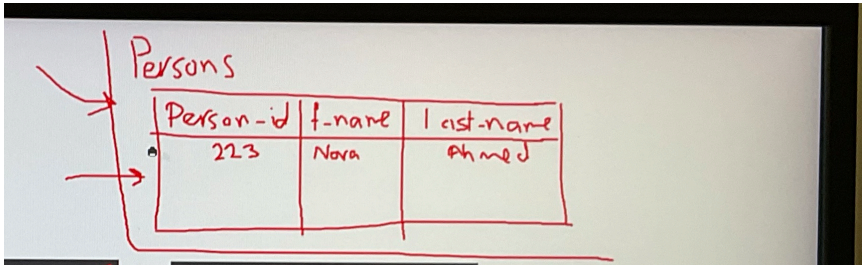
- DDL
- DML
- DCL
- TCL
- We will cover DDL in the next chapter, then DML in the one after.

DBMS LANGUAGES

Data Definition Language (DDL)

- It is used to define database structure or pattern.
- It is used to create schema, tables, indexes, constraints, etc. in the database.
- Using the DDL statements, you can create the skeleton of the database.
- Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.
- Create index, constraint, sequence

DDL COMMANDS




```sql
CREATE TABLE persons (
    person_id NUMBER,
    first_name VARCHAR2(50) NOT NULL,
    last_name VARCHAR2(50) NOT NULL,
    PRIMARY KEY(person_id)
);
```

```sql
CREATE INDEX members_last_name_i
ON members(last_name);
```

تغيير الجدول

```sql
ALTER TABLE members
ADD birth_date DATE NOT NULL;
```

```sql
DROP TABLE persons;
```

Select form

كل شيء ← Person

DBMS LANGUAGES

Data Manipulation Language (DML)

- It is used for accessing and manipulating data in a database. It handles user requests.
- Some DML Commands
  - Select: It is used to retrieve data from a database.
  - Insert: It is used to insert data into a table.
  - Update: It is used to update existing data within a table.
  - Delete: It is used to delete all records from a table.

DML COMMANDS

```
SELECT
  name
FROM
  customers;
```

NAME
Kimberly-Clark
Hartford Financial Services Group
Kraft Heinz
Fluor
AECOM
Jabil Circuit

```
SELECT
    customer_id,
    name,
    credit_limit
FROM
    customers;
```

CUSTOMER_ID	NAME	CREDIT_LIMIT
35	Kimberly-Clark	400
36	Hartford Financial Services Group	400
38	Kraft Heinz	500
40	Fluor	500
41	AECOM	500
44	Jabil Circuit	500

```
SELECT
    *
FROM
    orders
INNER JOIN order_items ON
    order_items.order_id = orders.order_id
ORDER BY
    order_date DESC;
```

ORDER_ID	CUSTOMER_ID	STATUS	SALESMAN_ID	ORDER_DATE	ITEM_ID	PRODUCT_ID	QUANTITY	UNIT_PRICE
88	6	Shipped	61	01-NOV-17	2	11	106	2015.11
88	6	Shipped	61	01-NOV-17	1	278	139	677.99
94	1	Shipped	62	27-OCT-17	6	4	111	2699.99
94	1	Shipped	62	27-OCT-17	5	181	143	999.99
94	1	Shipped	62	27-OCT-17	8	258	73	57.98
94	1	Shipped	62	27-OCT-17	4	80	133	564.89
94	1	Shipped	62	27-OCT-17	7	172	37	358.49
94	1	Shipped	62	27-OCT-17	2	186	146	1449.98
94	1	Shipped	62	27-OCT-17	3	218	86	1388.89
94	1	Shipped	62	27-OCT-17	9	12	33	824.98
94	1	Shipped	62	27-OCT-17	1	255	38	90.99
1	4	Pending	56	15-OCT-17	10	64	147	525.99

DBMS LANGUAGES

Data Control Language (DCL)

- DCL commands grant or revoke privileges to users.
- Some DCL Commands are:
  - **Grant:** It is used to give user access privileges to a database.
  - **Revoke:** It is used to take back permissions from the user.

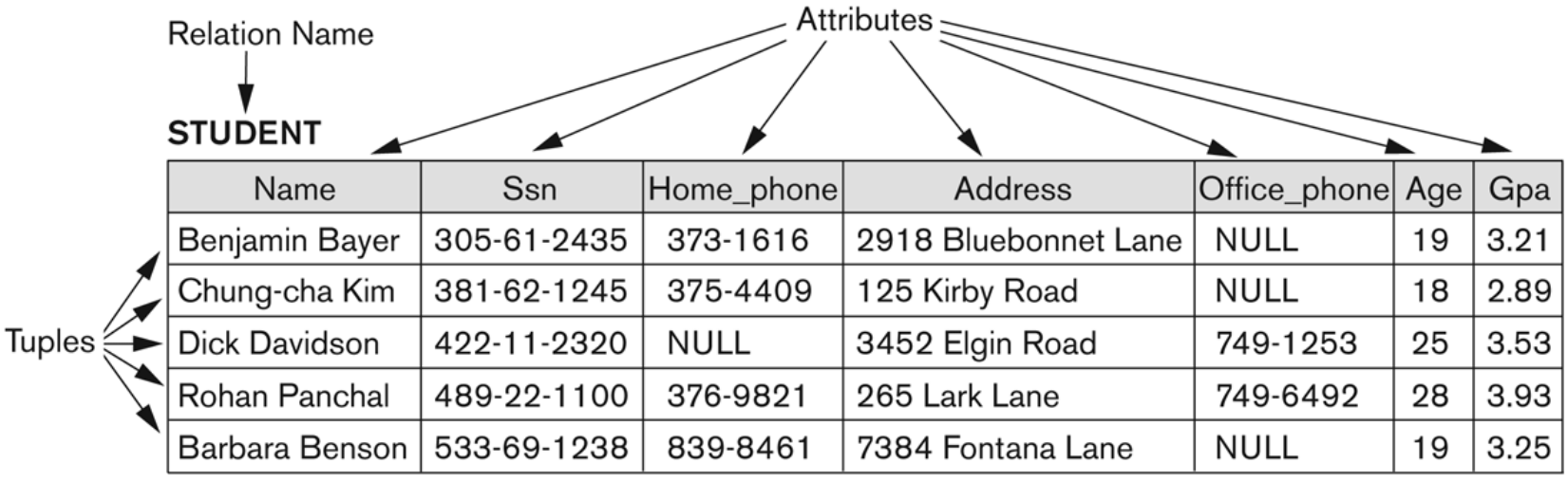
Transaction Control Language (TCL)

- TCL is used to run the changes made by the DML statement.
- Some TCL commands
  - **Commit:** It is used to save the transaction on the database.
  - **Rollback:** It is used to restore the database to original since the last Commit.

RELATIONAL MODEL COMPONENTS

- Relational database: A set of relations.
- Relation: 2 parts:
  - Schema: The name of relation, and name and type of attributes.
    - Example: *Employee (ID: string, name: string, dob: date).*
  - Instance: A table with rows (tuples) and columns.
    - Number of rows is cardinality.
    - Number of columns is degree or arity.
- Relational database schema: collection of all schemas.
- Database instance: collection of instances of all relations.

RELATION EXAMPLE



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1
The attributes and tuples of a relation STUDENT.

FORMAL DEFINITIONS

The **Schema** (or description) of a Relation:

- Denoted by R(A_1, A_2, ...A_n)
- R is the **name** of the relation
- The **attributes** of the relation are A_1, A_2, ..., A_n
- A_i is in D_i, which is the **domain** of the i-th attribute.
- A **domain** also contains null which is a special value that indicates unknown value.
- If A_1, ..., A_n are attributes with domains D_i,…, D_n then (A_1:D_1, …, A_n:D_n) is a **relation schema** that defines a relation type. Sometimes the domains are not included.

ACTIVITY

This table represents a relation R.

Answer the following questions:

1. What are the attributes?
2. What is the schema of R?
3. What are the tuples of R? Give one example.

R

A	B	C
0	1	2
3	4	5
6	7	9

DEFINITIONS SUMMARY

Informal Terms	**Formal Terms**
Table	Relation
Column Header	Attribute
All possible Column Values	Domain
Row	Tuple
Table Definition	Schema of a Relation
Populated Table	State of the Relation (or instance)

CHARACTERISTICS OF RELATIONS

Ordering of tuples in a relation r(R):

- The tuples are *not considered to be ordered*, even if they appear to be in the tabular form.

Ordering of attributes in a relation schema R (and of values within each tuple):

- We will consider the attributes in R(A1, A2, ..., An) and the values in t=<v1, v2, ..., vn> to be ordered .
  - (However, a more general alternative definition of relation does not require this ordering. It includes both the name and the value for each of the attributes).
  - Example: t= { <name, “John” >, <SSN, 123456789> }
  - This representation may be called as “self-describing”.

SAME STATE AS PREVIOUS FIGURE (BUT WITH DIFFERENT ORDER OF TUPLES)

Figure 5.2
The relation STUDENT from Figure 5.1 with a different order of tuples.

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21

CHARACTERISTICS OF RELATIONS

Values in a tuple:

- All values are considered atomic (indivisible).
- No composite or multivalued attributes are permitted. This property is known as the *First Normal Form*.
- Each value in a tuple must be from the domain of the attribute for that column
  - If tuple t = $<v_1, v_2, \ldots, v_n>$ is a tuple (row) in the relation instance r of $R(A_1, A_2, \ldots, A_n)$
  - Then each v_i must be a value from $dom(A_i)$
- A special **null** value is used to represent values that are unknown or not available or inapplicable in certain tuples.

CHARACTERISTICS OF RELATIONS

Notation:

- We refer to **component values** of a tuple t by:
  - $t[A_i]$ or $t.A_i$
  - This is the value v_i of attribute A_i for tuple t
- Similarly, $t[A_u, A_v, ..., A_w]$ refers to the sub-tuple of t containing the values of attributes $A_u, A_v, ..., A_w$, respectively in t

RELATIONAL MODEL CONSTRAINTS

Constraints determine which values are permissible and which are not in the database.

They are of three main types:

1. **Inherent Model-based or Implicit Constraints**: These are based on the data model itself. (e.g., relational model does not allow a list as a value for any attribute)
2. **Schema-based or Explicit Constraints**: They are expressed in the schema by using the facilities provided by the model. (e.g., max. cardinality ratio constraint in the ER model)
3. **Application based or semantic constraints**: These are constraints that cannot be directly expressed in the schemas of the data model, and hence must be expressed and enforced by the application programs.

INHERENT MODEL-BASED OR IMPLICIT CONSTRAINTS

Include:

- Ordering of tuples in a relation is not important
- Ordering of values within a tuple is important
- Values and nulls in the tuples e.g., a relation cannot have duplicate tuples is an inherent constraint.
- Interpretation (Meaning) of a Relation. Every relation represents facts about entities or relationships.

SCHEMA-BASED CONSTRAINTS

Constraints are **conditions** that must hold on **all** valid relation states.

There are three *main types* of (explicit schema-based) constraints that can be expressed in the relational model:

- **Key** constraints
- **Entity integrity** constraints
- **Referential integrity** constraints

Another schema-based constraint is the **domain** constraint

- Every value in a tuple must be from the *domain of its attribute* (or it could be **null**, if allowed for that attribute)

SCHEMA-BASED CONSTRAINTS

خن تذاكر

Include:

- Domain constraints.
- Key constraints.
- Constraints on nulls.
- Entity integrity constraints.
- Referential integrity constraints.

DOMAIN CONSTRAINTS

They specify that within each tuple, the value of each attribute must be an atomic value from the domain.

The data types associated with domains includes:

integers, real numbers, characters, Booleans, fixed length strings, variable length strings, date, time, timestamp...etc.

Other possible domains can be a range of values from a data type or as an enumerated data type in which all possible values are explicitly listed.

CONSTRAINTS ON NULL VALUES

Specifies whether NULL values are or are not permitted e.g.if every STUDENT tuple must have a valid, non-NULL value for the Name attribute, then the Name of STUDENT is constrained to be NOT NULL.

KEY CONSTRAINTS

Super-key of R:

- Is a set of attributes denoted by SK of schema R with the following condition:
  - No two tuples in any valid relation state r(R) will have the same value for SK
  - That is, for any distinct tuples t1 and t2 in r(R), t1[SK] ≠ t2[SK]
  - This condition must hold in *any valid state* r(R)

Key of R:

- A "minimal" super-key
- That is, a key is a super-key K such that removal of any attribute from K results in a set of attributes that is not a super-key (does not possess the super-key uniqueness property)

A Key is a Superkey but not vice versa

KEY CONSTRAINTS (CONTINUED)

Example: Consider the CAR relation schema:

- CAR(State, Reg#, SerialNo, Make, Model, Year)
- CAR has two keys:
  - Key1 = {State, Reg#}
  - Key2 = {SerialNo}
- Both are also superkeys of CAR
- {SerialNo, Make} is a superkey but *not* a key.

In general:

- Any *key* is a *superkey* (but not vice versa)
- Any set of attributes that *includes a key* is a *superkey*
- A *minimal* superkey is also a key

KEY CONSTRAINTS (CONTINUED)

If a relation has several **candidate keys**, one is chosen arbitrarily to be the **primary key**.

- The primary key attributes are underlined.

Example: Consider the CAR relation schema:

- CAR(State, Reg#, SerialNo, Make, Model, Year)
- We chose SerialNo as the primary key

The primary key value is used to *uniquely identify* each tuple in a relation

- Provides the tuple identity

Also used to *reference* the tuple from another tuple:

- General rule: Choose as primary key the smallest of the candidate keys (in terms of size)
- Not always applicable – choice is sometimes subjective

CAR TABLE WITH TWO CANDIDATE KEYS

CAR

License_number	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

Figure 5.4
The CAR relation, with two candidate keys: License_number and Engine_serial_number.

ACTIVITY

Department(Dept_num, Dept_name, Location, Manager_name)

- Write the key, candidate keys, and super-keys for the Department relation.
- Write an appropriate domain for each attribute of the Department relation.

RELATIONAL DATABASE SCHEMA

Relational Database Schema:

- A set S of relation schemas that belong to the same database.
- S is the name of the whole **database schema**
- S = {R1, R2, ..., Rn} and a set IC of integrity constraints.
- R1, R2, …, Rn are the names of the individual **relation schemas** within the database S

Following slide shows a COMPANY database schema with 6 relation schemas

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Figure 5.5
Schema diagram for the COMPANY relational database schema.

RELATIONAL DATABASE STATE

A **relational database state** DB of S is a set of relation states DB = $\{r_1, r_2, ..., r_m\}$ such that each r_i is a state of R_i and such that the r_i relation states satisfy the integrity constraints specified in IC.

A relational database *state* is sometimes called a relational database *snapshot* or *instance*.

We will not use the term *instance* since it also applies to single tuples.

A database state that does not meet the constraints is an invalid state

POPULATED DATABASE STATE

Each *relation* will have many tuples in its current relation state

The *relational database state* is a union of all the individual relation states

Whenever the database is changed, a new state arises

Basic operations for changing the database:

- INSERT a new tuple in a relation
- DELETE an existing tuple from a relation
- MODIFY an attribute of an existing tuple

Next slide (Fig. 5.6) shows an example state for the COMPANY database schema shown in Fig. 5.5.

Figure 5.6
One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

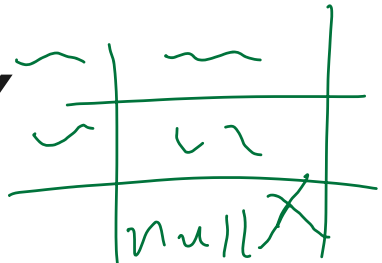
PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

ENTITY INTEGRITY



Entity Integrity:

- The *primary key attributes* PK of each relation schema R in S cannot have null values in any tuple of r(R).
  - This is because primary key values are used to *identify* the individual tuples.
  - t[PK] ≠ null for any tuple t in r(R)
  - If PK has several attributes, null is not allowed in any of these attributes
- Note: Other attributes of R may be constrained to disallow null values, even though they are not members of the primary key.

REFERENTIAL INTEGRITY

A constraint involving **two** relations

- The previous constraints involve a single relation.

Used to specify a **relationship** among tuples in two relations:

- The **referencing relation** and the **referenced relation**.

It is used to maintain the consistency among tuples in the two relations

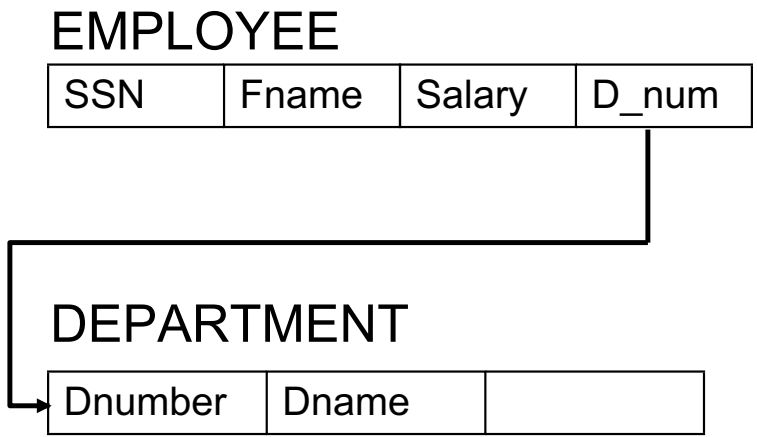
REFERENTIAL INTEGRITY

Tuples in the **referencing relation** R1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the **referenced relation** R2.

- A tuple t1 in R1 is said to **reference** a tuple t2 in R2 if t1[FK] = t2[PK].

A referential integrity constraint can be displayed in a relational database schema as a directed arc from R1.FK to R2.

REFERENTIAL INTEGRITY CONSTRAINTS



e.g. in figure, the attribute D_num of EMPLOYEE gives the department number for which each employee works; hence its value in every EMPLOYEE tuple must match the Dnumber value of some tuple in the DEPARTMENT relation.

REFERENTIAL INTEGRITY (OR FOREIGN KEY) CONSTRAINT

Statement of the constraint

- The value in the foreign key column (or columns) FK of the **referencing relation** R1 can be **either**:
  1. a value of an existing primary key value of a corresponding primary key PK in the **referenced relation** R2, or
  2. a **null**.

In case (2), the FK in R1 should **not** be a part of its own primary key.

DISPLAYING A RELATIONAL DATABASE SCHEMA AND ITS CONSTRAINTS

Each relation schema can be displayed as a row of attribute names

The name of the relation is written above the attribute names

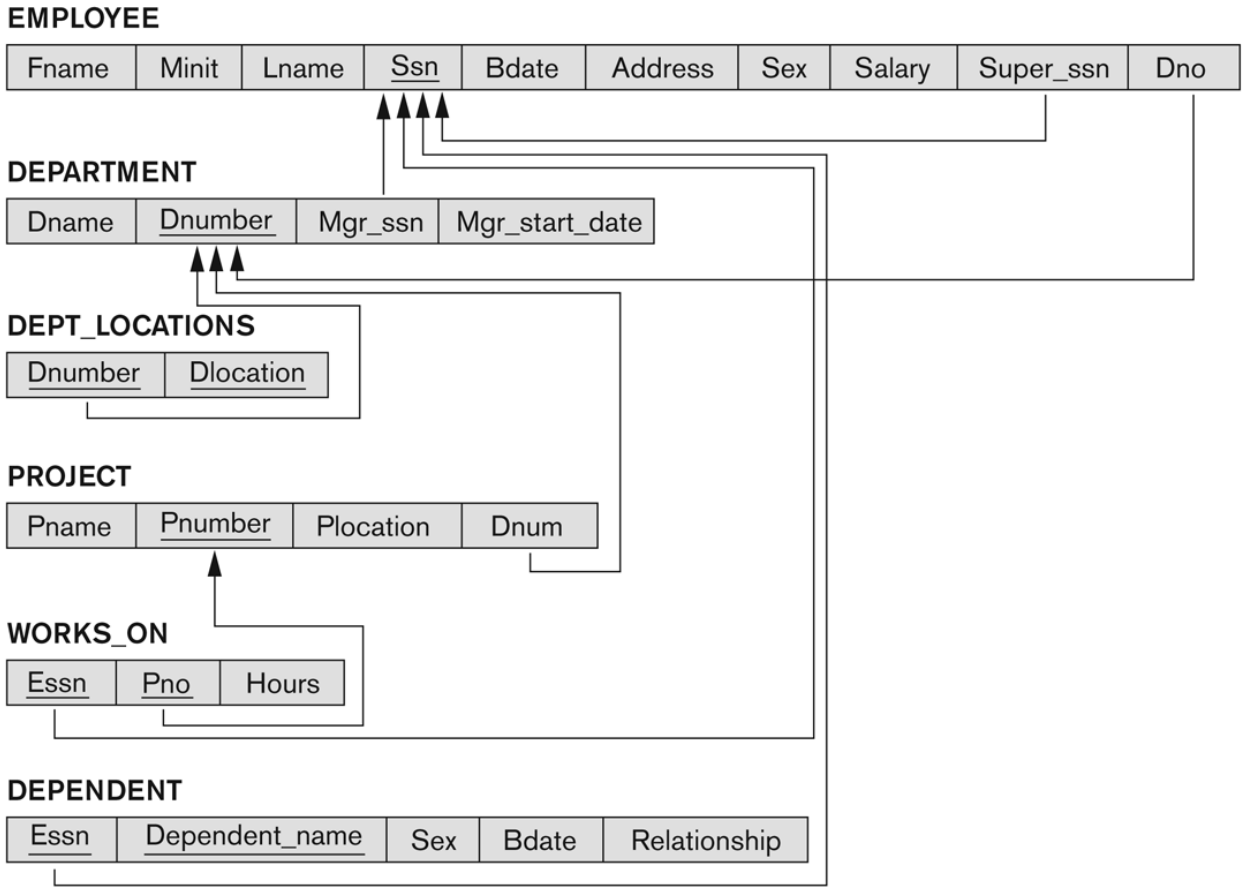
The primary key attribute (or attributes) will be underlined

A foreign key (referential integrity) constraints is displayed as a directed arc (arrow) from the foreign key attributes to the referenced table

- Can also point the the primary key of the referenced relation for clarity

Next slide shows the COMPANY **relational schema diagram with referential integrity constraints**

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.



OTHER TYPES OF CONSTRAINTS

```
CREATE ASSERTION assertion_name
CHECK (condition);
```

Semantic Integrity Constraints:

- based on application semantics and cannot be expressed by the model
- Example: “the max. no. of hours per employee for all projects he or she works on is 56 hrs per week”

A **constraint specification** language may have to be used to express these

SQL-99 allows **CREATE TRIGGER** and **CREATE ASSERTION** to express some of these semantic constraints

Keys, Permissibility of Null values, Candidate Keys (Unique in SQL), Foreign Keys, Referential Integrity etc. are expressed by the **CREATE TABLE** statement in SQL.

UPDATE OPERATIONS ON RELATIONS

INSERT a tuple.

DELETE a tuple.

MODIFY a tuple.

Integrity constraints should not be violated by the update operations.

Several update operations may have to be grouped together.

Updates may **propagate** to cause other updates automatically. This may be necessary to maintain integrity constraints.

UPDATE OPERATIONS ON RELATIONS

In case of integrity violation, several actions can be taken:

- Cancel the operation that causes the violation (RESTRICT or REJECT option)
- Perform the operation but inform the user of the violation
- Trigger additional updates so the violation is corrected (CASCADE option, SET NULL option)
- Execute a user-specified error-correction routine

POSSIBLE VIOLATIONS FOR EACH OPERATION

Cases

INSERT may violate any of the constraints:

- Domain constraint:
  - if one of the attribute values provided for the new tuple is not of the specified attribute domain
- Key constraint:
  - if the value of a key attribute in the new tuple already exists in another tuple in the relation
- Referential integrity:
  - if a foreign key value in the new tuple references a primary key value that does not exist in the referenced relation
- Entity integrity:
  - if the primary key value is null in the new tuple

POSSIBLE VIOLATIONS FOR EACH OPERATION

DELETE may violate only referential integrity:

- If the primary key value of the tuple being deleted is referenced from other tuples in the database
  - Can be remedied by several actions: RESTRICT, CASCADE, SET NULL (see Chapter 6 for more details)
    - RESTRICT option: reject the deletion
    - CASCADE option: propagate the new primary key value into the foreign keys of the referencing tuples
    - SET NULL option: set the foreign keys of the referencing tuples to NULL
- One of the above options must be specified during database design for each foreign key constraint.
- on delete set null.
- on delete cascade

THE DELETE OPERATION

Delete the WORKS_ON tuple with Essn ='999887777' and Pno = 10

The seletion is acceptable and deletes exactly one tuple

Delete the EMPLOYEE tuple with Ssn = '999887777'

This deletion is unacceptable, because there are tuples in WORK_ON that refer to this tuple. Hence if the tuple in EMPLOYEE is deleted, referential integrity violations will result.

Delete the EMPLOYEE tuple with Ssn = '333445555'

The deletion is unacceptable because the tuple involved is referenced by tuples from the EMPLOYEE, DEPARTMENT, WORKS_ON and DEPENDENT relation.

POSSIBLE VIOLATIONS FOR EACH OPERATION

UPDATE may violate domain constraint and NOT NULL constraint on an attribute being modified

Any of the other constraints may also be violated, depending on the attribute being updated:

- Updating the primary key (PK):
  - Similar to a DELETE followed by an INSERT
  - Need to specify similar options to DELETE
- Updating a foreign key (FK):
  - May violate referential integrity
- Updating an ordinary attribute (neither PK nor FK):
  - Can only violate domain constraints

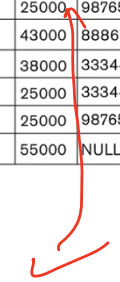
DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

THE UPDATE OPER



```
Update the salary of the EMPLOYEE tuple with Ssn ='999887777'
to 28000
```

This query is acceptable

```
Update the D_num of the EMPLOYEE tuple with Sssn ='999887777'
to 1
```

Acceptable

```
Update the Ssn of the EMPLOYEE in the tuple with
Ssn='999887777' to '987654321'
```

Unacceptable, because it violates primary key constraint by repeating a value that already exist as a primary key in another tuple; it violates referential integrity constraints because there are other relations that refer to the existing value of Ssn.

ACTIVITY

Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course:

STUDENT(SSN, Name, Major, Bdate)

COURSE(Course#, Cname, Dept)

ENROLL(SSN, Course#, Quarter, Grade)

BOOK_ADOPTION(Course#, Quarter, Book_ISBN)

TEXT(Book_ISBN, Book_Title, Publisher, Author)

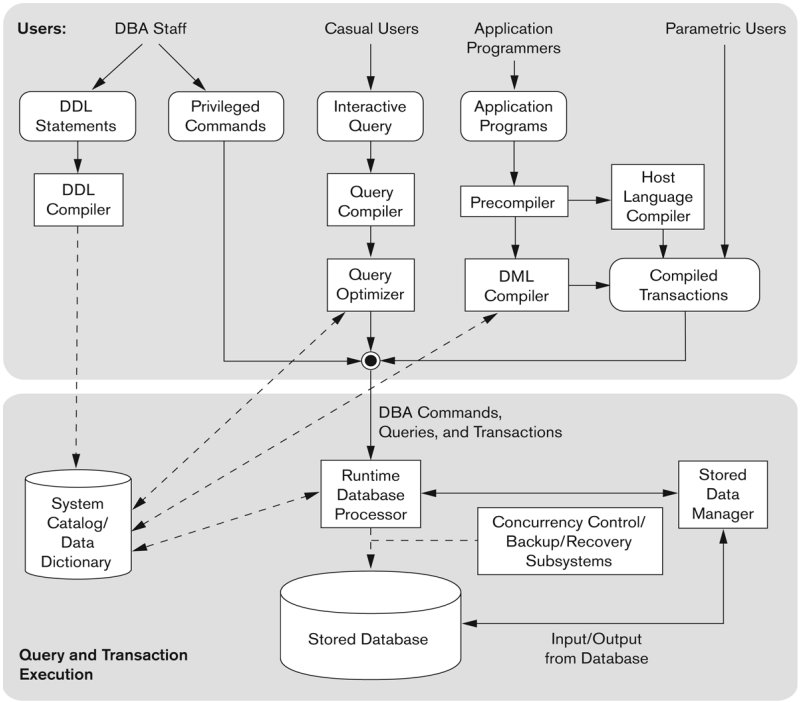
Draw a relational schema diagram specifying the foreign keys for this schema.

NEXT..

We have also looked at all possible constraints in the database.

We will look at some DBMS components next.

TYPICAL DBMS COMPONENT MODULES



DBMS COMPONENT MODULES

The database and DBMS catalog are usually stored on disk.

Access to the disk is controlled by the operating system (OS) which schedules disk read/write.

A higher-level, stored data manager module (component) controls access to DBMS information that is stored on disk, whether it is part of the database or the catalog.

DBMS has interfaces for the DBA staff who work on defining the database and tuning it by making changes to its definition using the DDL and other privileged commands.

DBMS COMPONENT MODULES

The **query compiler** handles high-level queries that are entered interactively. These queries are parsed and validated for correctness of the query syntax, the names of files and data elements, and so on by a query compiler that compiles them into an internal form, which is then subjected to query optimization.

The **query optimizer** is concerned with rearrangement and possible reordering of operations, elimination of redundancies, and use of efficient search algorithms during execution. It consults the system catalog for statistical and other physical information about the stored data and generates executable code that performs the necessary operations for the query and makes calls on the runtime processor.

CENTRALIZED AND CLIENT-SERVER DBMS ARCHITECTURES

Centralized DBMS:

- Combines everything into single system including- DBMS software, hardware, application programs, and user interface processing software.
- User can still connect through a remote terminal – however, all processing is done at centralized site.

A PHYSICAL CENTRALIZED ARCHITECTURE

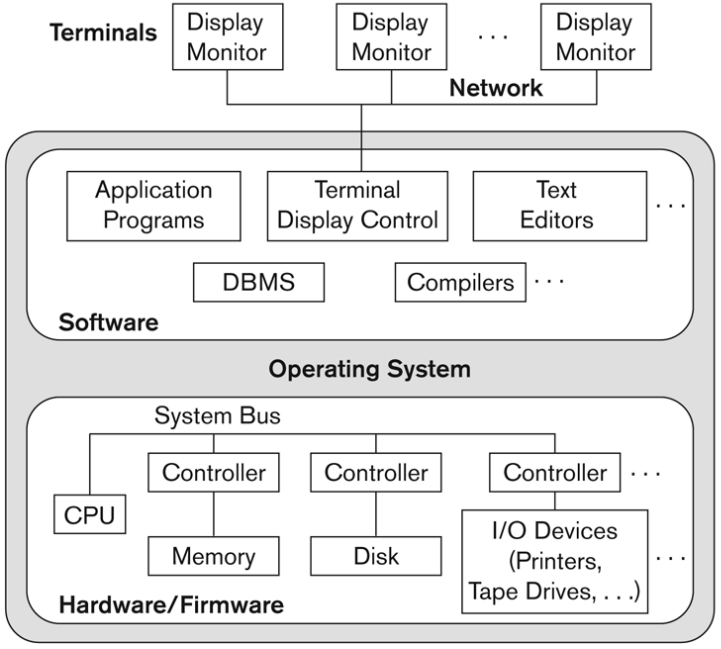


Figure 2.4
A physical centralized architecture.

BASIC 2-TIER CLIENT-SERVER ARCHITECTURES

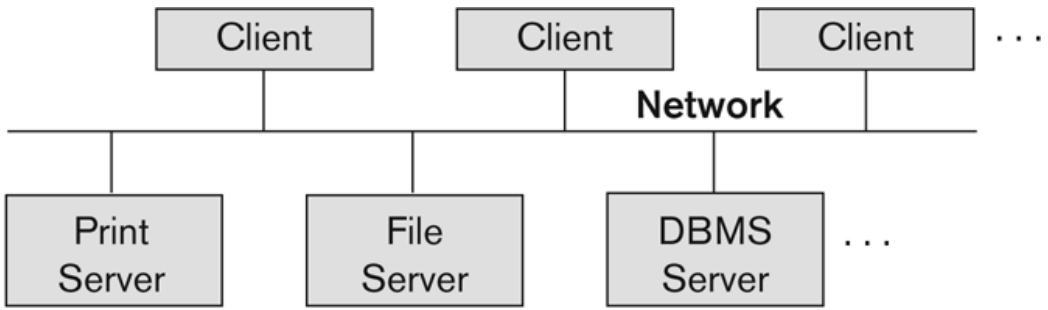
Specialized Servers with Specialized functions

- Print server
- File server
- DBMS server
- Web server
- Email server

Clients can access the specialized servers as needed

PHYSICAL TWO-TIER CLIENT SERVER ARCHITECTURE

Figure 2.5
Logical two-tier client/server architecture.



CLIENTS

Provide appropriate interfaces through a client software module to access and utilize the various server resources.

Clients may be diskless machines or PCs or Workstations with disks with only the client software installed.

Connected to the servers via some form of a network.

(LAN: local area network, wireless network, etc.)

DBMS SERVER

Provides database query and transaction services to the clients

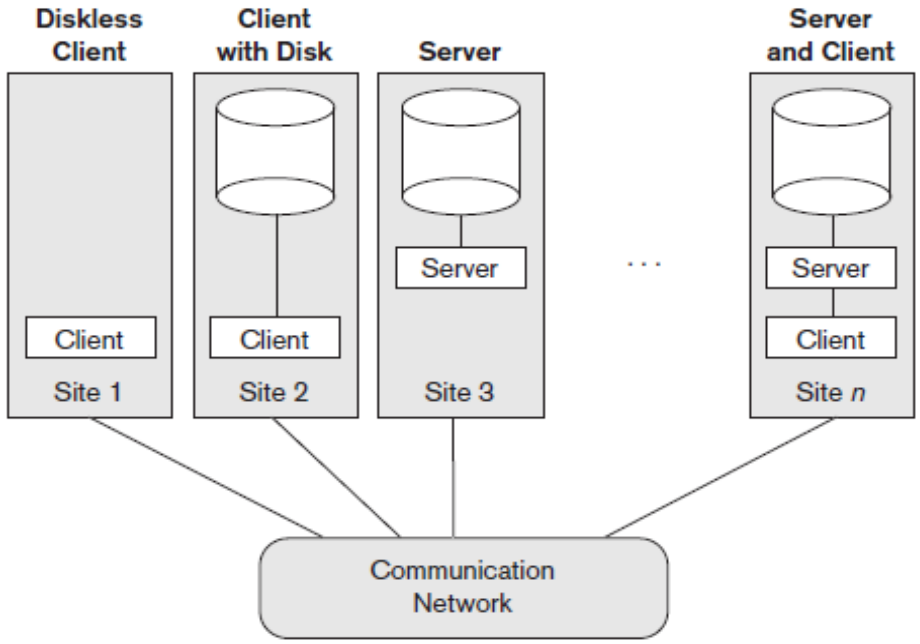
Relational DBMS servers are often called SQL servers, query servers, or transaction servers

Applications running on clients utilize an Application Program Interface (**API**) to access server databases via standard interface such as:

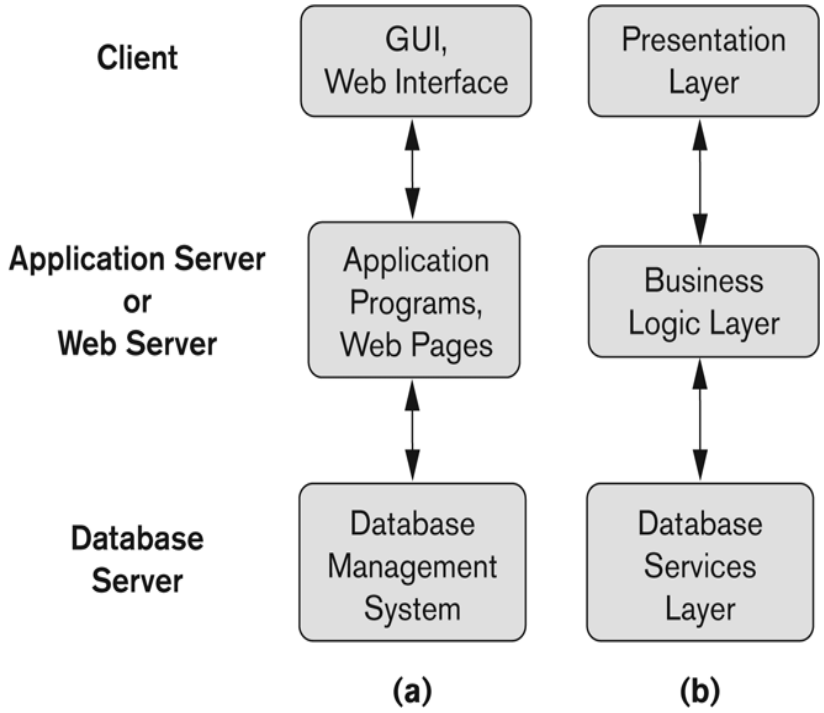
- ODBC: Open Database Connectivity standard
- JDBC: for Java programming access

LOGICAL TWO-TIER CLIENT SERVER ARCHITECTURE

Figure 2.6
Physical two-tier client/server architecture.



THREE-TIER CLIENT-SERVER ARCHITECTURE



The **presentation** layer displays information to the user and allows data entry.

The **business logic** layer handles intermediate rules and constraints before data is passed up to the user or down to the DBMS.

The **database services** layer includes all data management services.

THREE TIER CLIENT-SERVER ARCHITECTURE

The emergence of web changed the roles of client and server. Common for Web applications

Three-tier architecture is a well-established software application architecture that organizes applications into three logical and physical computing tiers: the presentation tier, or user interface; the application tier, where data is processed; and the data tier, where the data associated with the application is stored and managed.

The chief benefit of three-tier architecture is that because each tier runs on its own infrastructure, each tier can be developed simultaneously by a separate development team, and can be updated or scaled as needed without impacting the other tiers.

THREE TIER CLIENT-SERVER ARCHITECTURE

Intermediate Layer called Application Server or Web Server:

- Stores the web connectivity software and the business logic part of the application used to access the corresponding data from the database server
- Acts like a conduit for sending partially processed data between the database server and the client.

Three-tier Architecture Can Enhance Security:

- Database server only accessible via middle tier
- Clients cannot directly access database server
- Clients contain user interfaces and Web browsers
- The client is typically a PC or a mobile device connected to the Web

THREE TIER CLIENT-SERVER ARCHITECTURE

Faster development: Because each tier can be developed simultaneously by different teams, an organization can bring the application to market faster, and programmers can use the latest and best languages and tools for each tier.

Improved scalability: Any tier can be scaled independently of the others as needed.

Improved reliability: An outage in one tier is less likely to impact the availability or performance of the other tiers.

Improved security: Because the presentation tier and data tier can't communicate directly, a well-designed application tier can function as a sort of internal firewall, preventing SQL injections and other malicious exploits.

SUMMARY

Presented Relational Model Concepts
- Definitions
- Characteristics of relations

Discussed Relational Model Constraints and Relational Database Schemas
- Domain constraints
- Key constraints
- Entity integrity
- Referential integrity

Described the Relational Update Operations and Dealing with Constraint Violations