Giới thiệu môn học

Bổ túc kiến thức Nhập Môn CSDL

Tuần 1: Giới thiệu hệ cơ sở dữ liệu – mô hình dữ liệu quan hệ – SQL căn bản

Tuần 2: SQL nâng cao – Bài tập SQL

Tuần 3: Mô hình quan hệ thực thể ER - và mô hình mở rộng EER

Tuần 4: Ánh xạ dữ liệu quan hệ thực thể EER – Bài tập

Tuần 5: Phụ thuộc hàm – Chuẩn hoá CSDL quan hệ – Bài tập

Tuần 6: View-Trigger-Procedure- Ôn tập

Website: www.cse.hcmut.edu.vn/~ttqnguyet

Chapter 1

Introduction to Database System

Lecturer: Ms. Tran Thi Que Nguyet
Email: ttqnguyet@cse.hcmut.edu.vn
quenguyettran@gmail.com

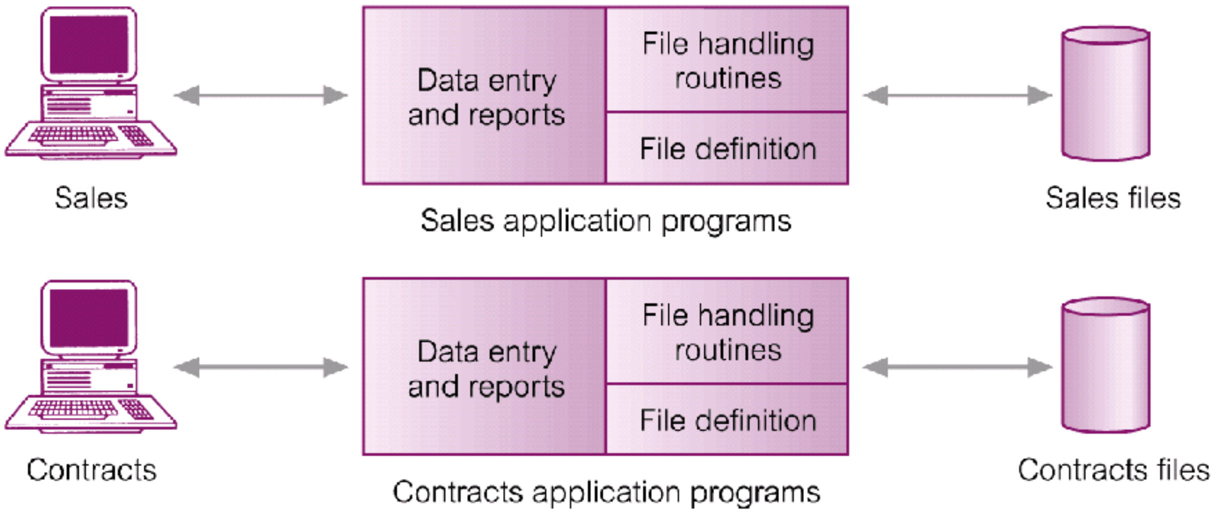
Outline

- Introduction
  - File based approach
  - Database approach
  - Basic definitions
- Database systems concepts
  - Data model
  - Three schema architecture – Data independence
  - Database schema – state – instance
  - DBMS languages
  - Classification of DBMS
  - Database users

File-based Approach

- Data is stored in one or more separate computer files
- Data is then processed by computer programs - **applications**

File-based Approach



Sales Files

PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo)

PrivateOwner (ownerNo, fName, lName, address, telNo)

Client (clientNo, fName, lName, address, telNo, prefType, maxRent)

Contracts Files

Lease (leaseNo, propertyNo, clientNo, rent, paymentMethod, deposit, paid, rentStart, rentFinish, duration)

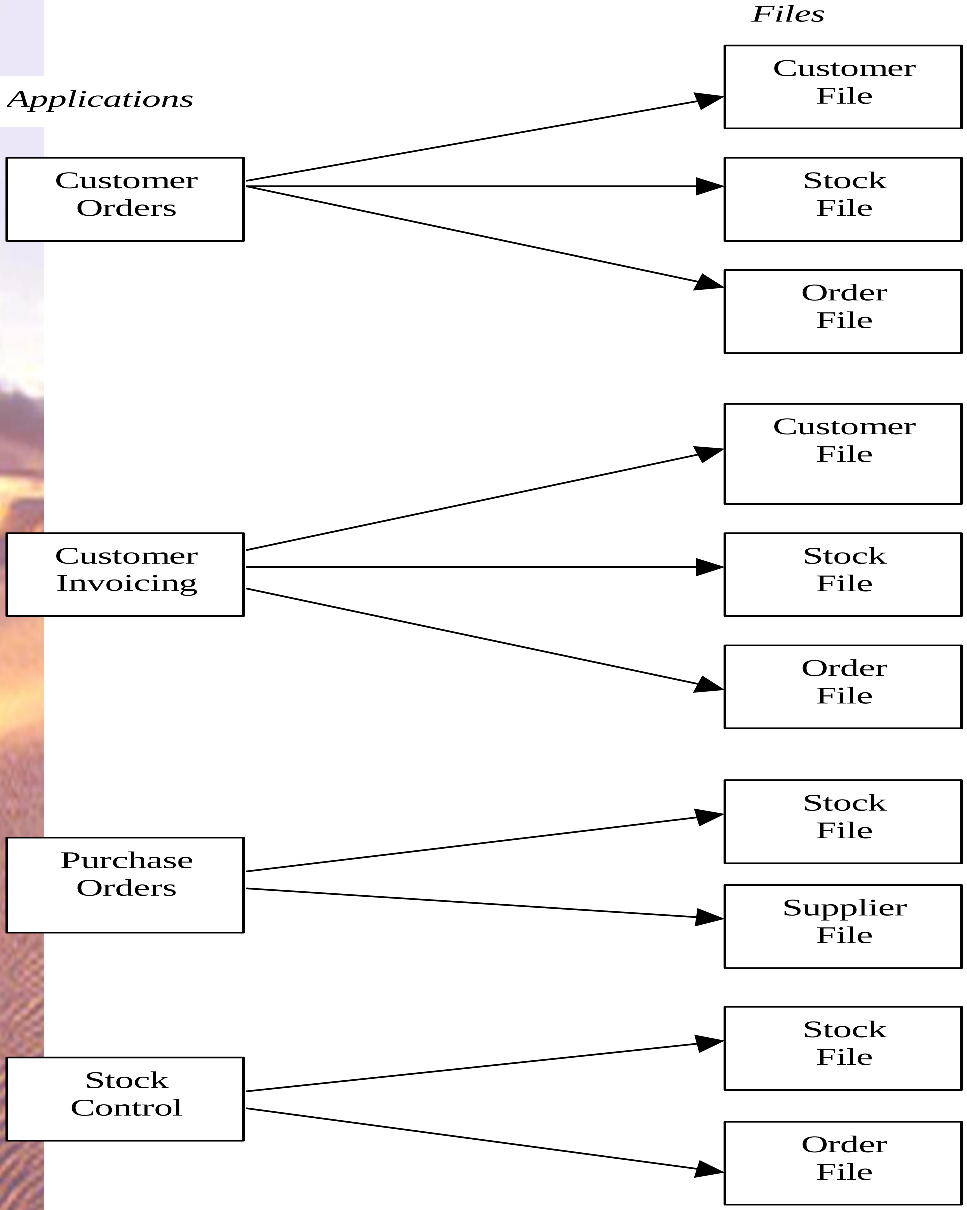
PropertyForRent (propertyNo, street, city, postcode, rent)

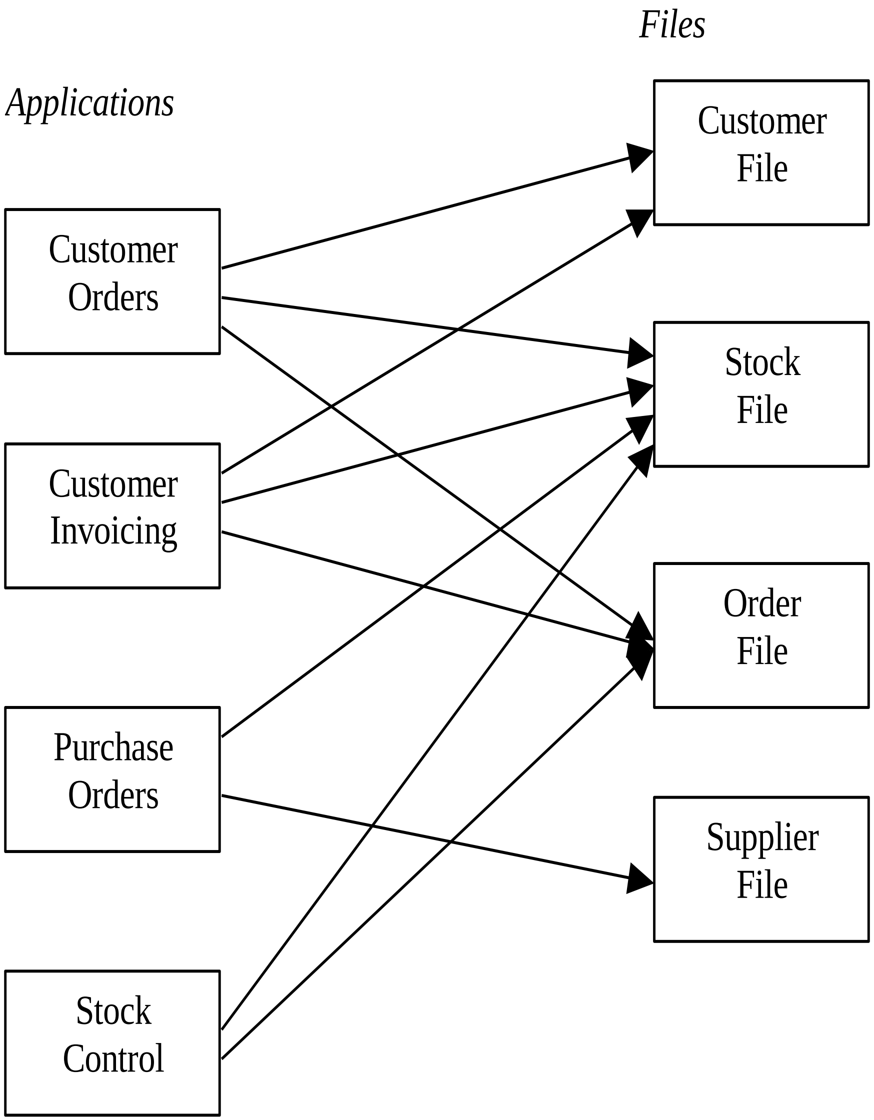
Client (clientNo, fName, lName, address, telNo)

File-based Approach

- Problems/Limitations
  - Data Redundancy
  - Data Inconsistency
  - *More details: see [2]*

File-based approach





Shared file approach

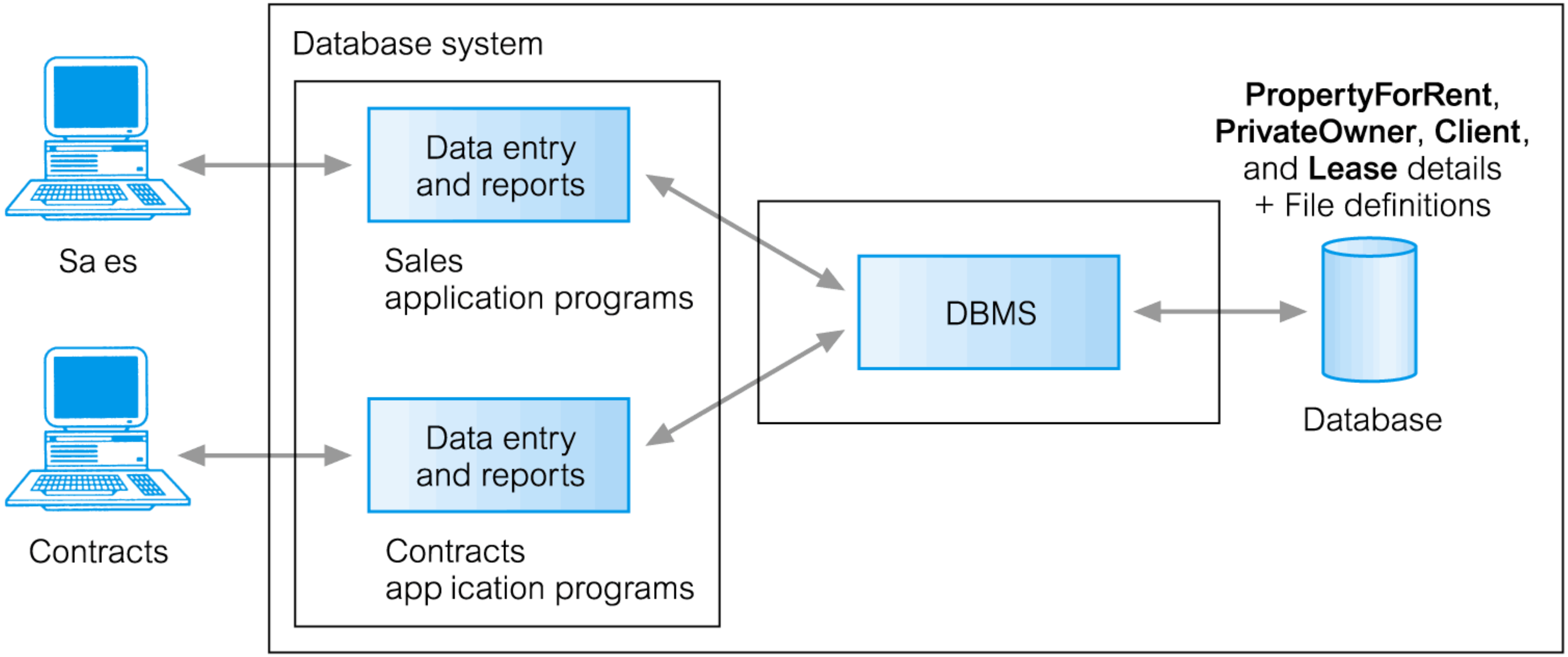
File-based Approach

- **Shared File Approach**
  - Data (files) is shared between different applications
  - Data redundancy problem is alleviated
  - Data inconsistency problem across different versions of the same file is solved
  - Other problems:
    - Rigid data structure: If applications have to share files, the file structure that suits one application might not suit another
    - Physical data dependency: If the structure of the data file needs to be changed in some way, this alteration will need to be reflected in all application programs that use that data file
    - No support of concurrency control: While a data file is being processed by one application, the file will not be available for other applications or for ad hoc queries

Database Approach

- Arose because:
  - Definition of data was embedded in application programs, rather than being stored separately and independently
  - No control over access and manipulation of data beyond that imposed by application programs
- Result:
  - The Database and Database Management System (DBMS).

Database Approach



PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo)
PrivateOwner (ownerNo, fName, lName, address, telNo)
Client (clientNo, fName, lName, address, telNo, prefType, maxRent)
Lease (leaseNo, propertyNo, clientNo, paymentMethod, deposit, paid, rentStart, rentF nish)

Basic Definitions

- **Database**: A collection of related data.
- **Data**: Known facts that can be recorded and have an implicit meaning.
- **Mini-world**: Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.
- **Database Management System (DBMS)**: A software package/ system to facilitate the creation and maintenance of a computerized database.
- **Database System**: The DBMS software together with the data itself. Sometimes, the applications are also included.

Typical DBMS Functionality

- Define a database : in terms of data types, structures and constraints
- Construct or Load the Database on a secondary storage medium
- Manipulating the database : querying, generating reports, insertions, deletions and modifications to its content
- Concurrent Processing and Sharing by a set of users and programs – yet, keeping all data valid and consistent

Typical DBMS Functionality

- Other features:
  - Protection or Security measures to prevent unauthorized access
  - “Active” processing to take internal actions on data
  - Presentation and Visualization of data

Example of a Database

- **Mini-world for the example**: Part of a UNIVERSITY environment.
- **Some mini-world *entities***:
  - STUDENTs
  - COURSEs
  - SECTIONs (of COURSEs)
  - (academic) DEPARTMENTs
  - INSTRUCTORs

Note: The above could be expressed in the ENTITY-RELATIONSHIP data model.

Example of a Database

- **Some mini-world *relationships*:**
  - SECTIONs *are of* specific COURSEs
  - STUDENTs *take* SECTIONs
  - COURSEs *have* prerequisite COURSEs
  - INSTRUCTORs *teach* SECTIONs
  - COURSEs *are offered by* DEPARTMENTs
  - STUDENTs *major in* DEPARTMENTs

Note: The above could be expressed in the *ENTITY-RELATIONSHIP* data model.

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
....	....	.....
....	....	.....
....	....	.....
Prerequisite_number	XXXXNNNN	PREREQUISITE

Note: Major_type is defined as an enumerated type with all known majors.
XXXXNNNN is used to define a type with four alpha characters followed by four digits.

Main Characteristics of the Database Approach

- Self-describing nature of a database system:
  - Contains catalog (metadata)
- Insulation between programs and data: (**program-data independence**)

  Allows changing data storage structures and operations without having to change the DBMS access programs.

Main Characteristics of the Database Approach

- Data Abstraction: A **data model** is used to hide storage details and present the users with a *conceptual view* of the database.
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing

The Transaction Concept

- **Transaction**
  - Executing program
  - Includes some database operations
  - Must leave the database in a valid or consistent state
- **Online transaction processing (OLTP) systems**
  - Execute transactions at rates that reach several hundred per second

Outline

- Introduction
  - File based approach
  - Database approach
  - Basic definitions
- **Database systems concepts**
  - Data models
  - Three schema architecture – Data independence
  - Database schema – state – instance
  - DBMS languages
  - Classification of DBMS
  - Database users

Data Models

- **Data Model**: A set of concepts to describe the *structure* of a database, and certain *constraints* that the database should obey.
- **Data Model Operations**: Operations for specifying database retrievals and updates by referring to the concepts of the data model. Operations on the data model may include *basic operations* and *user-defined operations*.

Categories of data models

- **Conceptual** (**high-level**, **semantic**) data models: Provide concepts that are close to the way many users *perceive* data. (Also called **entity-based** or **object-based** data models.)
- **Physical** (**low-level**, **internal**) data models: Provide concepts that describe details of how data is stored on the computer storage media
- **Implementation** (**representational**) data models: Provide concepts that fall between the above two, balancing user views with some computer storage details.

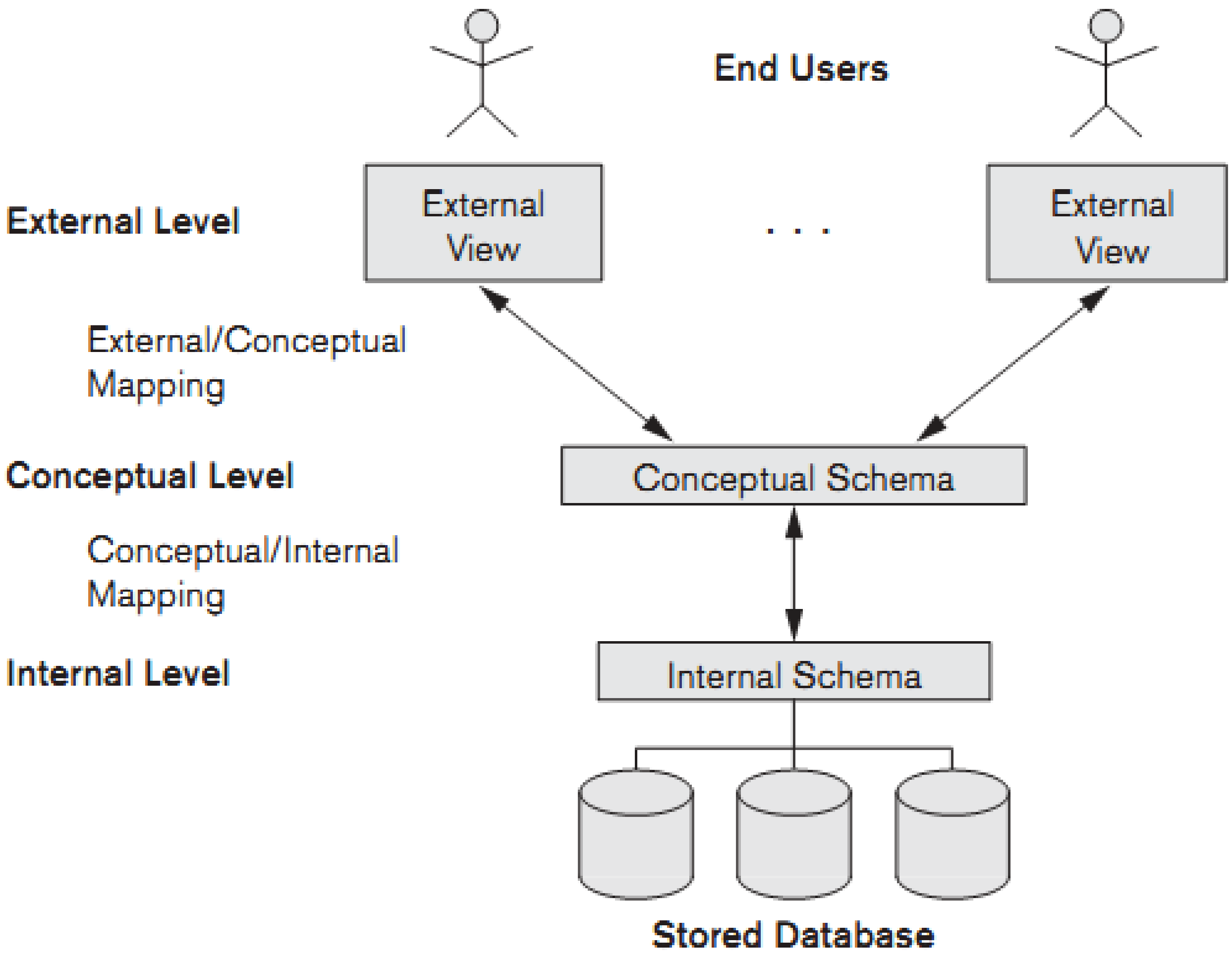
Three-Schema Architecture

- Proposed to support DBMS characteristics of:
  - **Program-data independence**.
  - Support of **multiple views** of the data.

Three-Schema Architecture

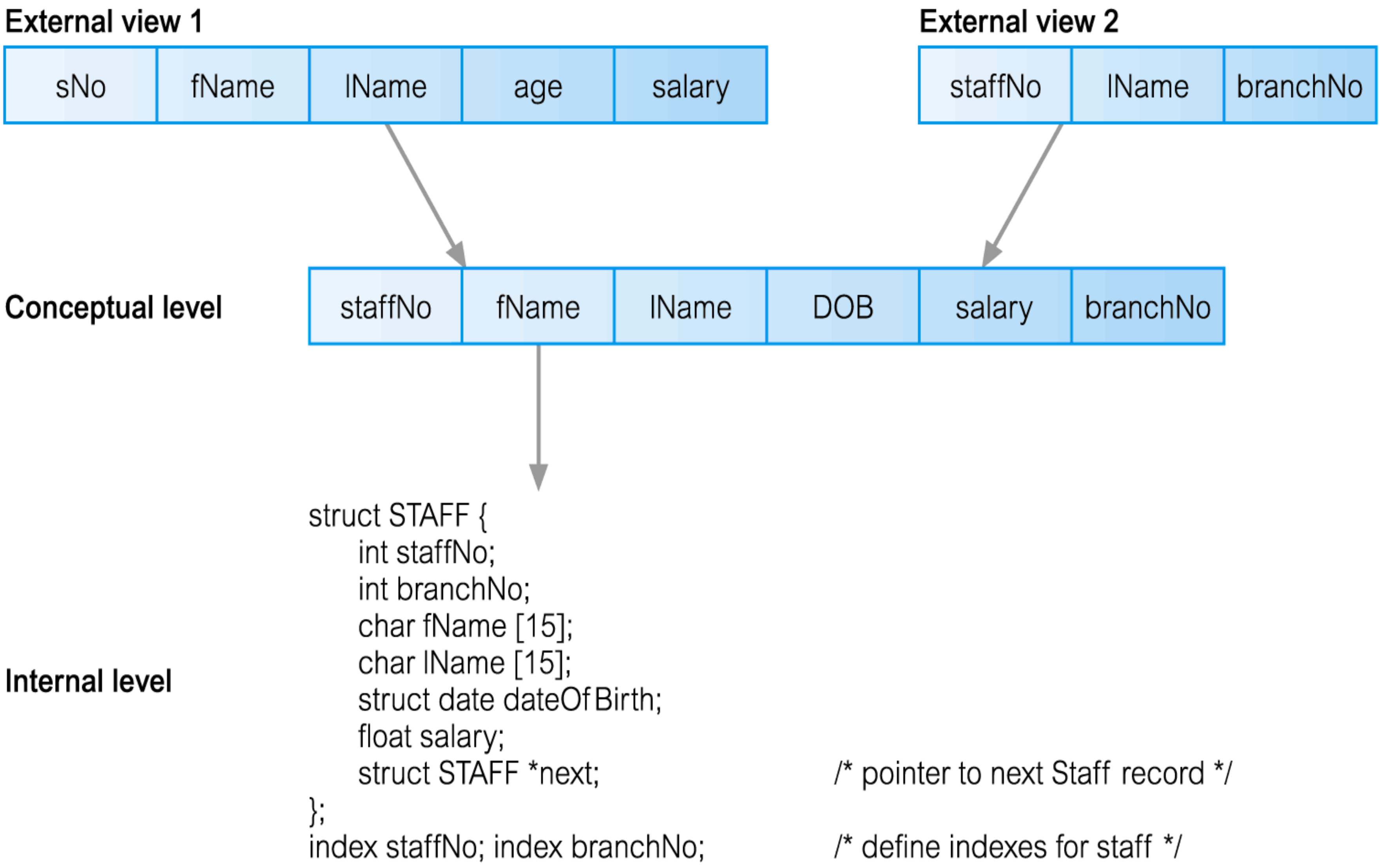
- **Objectives of Three-Schema Architecture**
  - All users should be able to access same data
  - A user's view is immune to changes made in other views
  - Users should not need to know physical database storage details
  - DBA should be able to change database storage structures without affecting the users' views
  - Internal structure of database should be unaffected by changes to physical aspects of storage
  - DBA should be able to change conceptual structure of database without affecting all users

Three-Schema Architecture



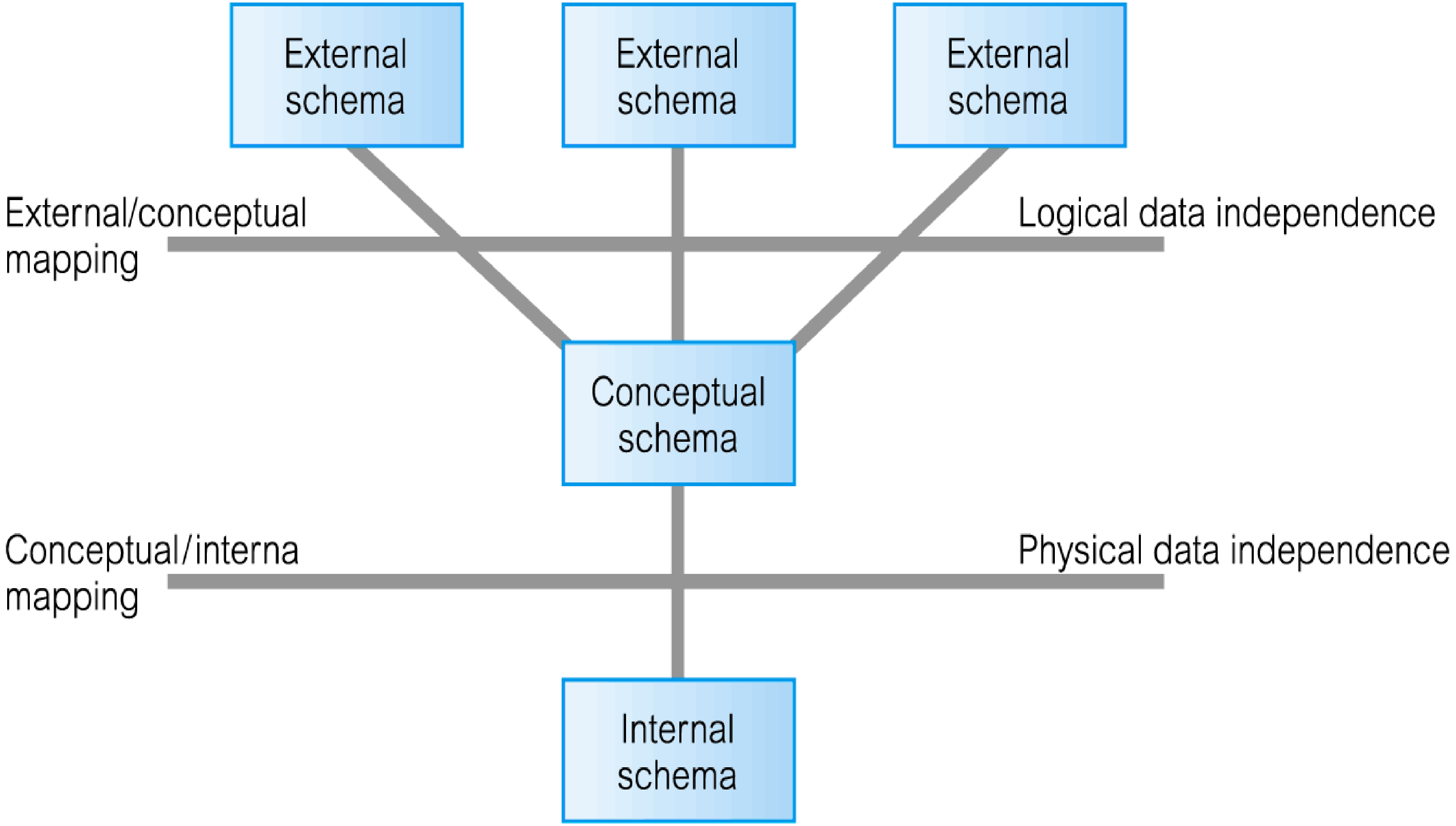
Three-Schema Architecture

- Defines DBMS schemas at *three levels*:
  - **Internal schema** at the internal level to describe physical storage structures and access paths. Typically uses a *physical* data model.
  - **Conceptual schema** at the conceptual level to describe the structure and constraints for the *whole* database for a community of users. Uses a *conceptual* or an *implementation* data model.
  - **External schemas** at the external level to describe the various user views. Usually uses the same data model as the conceptual level.


External view 1
sNo
fName
lName
age
salary
External view 2
staffNo
lName
branchNo
Conceptual level
staffNo
fName
lName
DOB
salary
branchNo
Internal level
struct STAFF {
int staffNo;
int branchNo;
char fName [15];
char lName [15];
struct date dateOfBirth;
float salary;
struct STAFF *next;
/* pointer to next Staff record */
};
index staffNo; index branchNo;
/* define indexes for staff */

Data Independence

- Data Independence is the capacity to change the schema at one level of a database system without having to change the schema at the next higher level
- **Logical Data Independence**: Change *conceptual schema* without having to change *external schemas* and their application programs.
- **Physical Data Independence**: Change *internal schema* without having to change *conceptual schema*.


External schema
External schema
External schema
External/conceptual mapping
Logical data independence
Conceptual schema
Conceptual/interna mapping
Physical data independence
Internal schema

Historical Development of Database Technology

- **Early Database Applications:** Hierarchical and Network Models (in mid 1960's).
- **Relational Model based Systems:** Researched and experimented with in IBM and the universities (in 1970).

Read more [2]

- **Object-oriented applications:** OODBMSs (in late 1980's and early 1990's)
- **Data on the Web and E-commerce Applications:** using new standards like XML (eXtended Markup Language).

Schemas versus Instances

- **Database Schema**: The *description* of a database.
- **Schema Diagram**: A diagrammatic display of (some aspects of) a database schema.
- **Schema Construct**: A component of the schema or an object within the schema, e.g., STUDENT, COURSE.
- **Database Instance**: The actual data stored in a database at a *particular moment in time*. Also called **database state** (or **occurrence**).

Database Schema Vs. Database State

- **Database State:** Refers to the content of a database at a moment in time.
- **Initial Database State:** Refers to the database when it is loaded
- **Valid State:** A state that satisfies the structure and constraints of the database.
- **Distinction**
  - The **database schema** changes *very infrequently*. The **database state** changes *every time the database is updated.*
  - **Schema** is also called **intension**, whereas **state** is called **extension**.

DBMS Languages

- **Data Definition Language (DDL)** allows the DBA or user to describe and name entities, attributes, and relationships required for the application plus any associated integrity and security constraints
- **Data Manipulation Language (DML)** provides basic data manipulation operations on data held in the database
- **Data Control Language (DCL)** defines activities that are not in the categories of those for the DDL and DML, such as granting privileges to users, and defining when proposed changes to a databases should be irrevocably made

DBMS Languages

- **Low Level or Procedural DML:** allow user to tell system exactly how to manipulate data (e.g., Network and hierarchical DMLs, example: GET UNIQUE, GET NEXT, GET NEXT WITHIN PARENT, etc.)
- **High Level** or **Non-procedural DML**(declarative language): allow user to state what data is needed rather than how it is to be retrieved (e.g., SQL, QBE)

Classification of DBMSs

- **Based on the data model used**:
  - Traditional: Relational, Network, Hierarchical.
  - Emerging: Object-oriented, Object-relational.
- **Other classifications:**
  - Single-user (typically used with micro-computers) vs. multi-user (most DBMSs).
  - Centralized (uses a single computer with one database) vs. distributed (uses multiple computers, multiple databases)

Database Users

- Users may be divided into:
  - Those who actually use and control the content (called "Actors on the Scene").
  - Those who enable the database to be developed and the DBMS software to be designed and implemented (called "Workers Behind the Scene").

Actors on the Scene

- **Database administrators (DBA)** are responsible for:
  - Authorizing access to the database
  - Coordinating and monitoring its use
  - Acquiring software and hardware resources
- **Database designers** are responsible for:
  - Identifying the data to be stored
  - Choosing appropriate structures to represent and store this data

Actors on the Scene (cont'd.)

- **End users**
  - People whose jobs require access to the database
  - Types:
    - **Casual end users:** use database occasionally, needing different information each time; use query language to specify their requests; typically middle- or high-level managers.
    - **Naive/Parametric end users**: Typically the biggest group of users; frequently query/update the database using standard **canned transactions** that have been carefully programmed and tested in advance.
    - **Sophisticated end users**: engineers, scientists, business analysts who implement their own applications to meet their complex needs.
    - **Stand-alone users**: Use "personal" databases, possibly employing a special-purpose (e.g., financial) software package.

Actors on the Scene (cont'd.)

- **System Analysts**: determine needs of end users, especially naive and parametric users, and develop specifications for canned transactions that meet these needs.
- **Application Programmers**: Implement, test, document, and maintain programs that satisfy the specifications mentioned above.

Workers behind the Scene

- **DBMS system designers and implementers**
  – Design and implement the DBMS modules and interfaces as a software package
- **Tool developers**
  – Design and implement **tools**
- **Operators and maintenance personnel**
  – Responsible for running and maintenance of hardware and software environment for database system

Summary

- We will study:
  - How to design a database
  - How to implement a database into DBMS
  - How to manipulate in a database system
  - How to prevent unauthorized accesses

Chapter 2

The Relational Data Model & SQL

Outline

- Relational Model Concepts
- Relational Model Constraints and Relational Database Schemas
- Update Operations and Dealing with Constraint Violations
- Basic SQL

Relational Model Concepts

- The model was first proposed by Dr. E.F. Codd of IBM in 1970 in the following paper:
  "A Relational Model for Large Shared Data Banks," Communications of the ACM, June 1970.

  The above paper caused a major revolution in the field of Database management and earned Ted Codd the coveted ACM Turing Award.

INFORMAL DEFINITIONS

- Represents data as a collection of relations
- RELATION: A table of values
  - A relation may be thought of as a **set of rows**.
  - A relation may alternately be though of as a **set of columns**.
  - Each row represents a fact that corresponds to a real-world **entity** or **relationship**.
  - Each row has a value of **an item or set of items** that **uniquely identifies** that row in the table.
  - Sometimes **row-ids** or **sequential numbers** are assigned to identify the rows in the table.
  - Each column typically is called by its column name or column header or attribute name.

PROJECT

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

EMPLOYEE

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

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

FORMAL DEFINITIONS

- A **Relation** may be defined in multiple ways.
- The **Schema** of a Relation: *R* (A1, A2, ...An)

  Relation schema *R* is defined over **attributes** A1, A2, ...An
- A relation (or relation state) r = {t1, t2, t3, .., tm} (m tuples)

Exp: CUSTOMER (Cust-id, Cust-name, Address, Phone#)

FORMAL DEFINITIONS

- A **tuple** t is an ordered set of values

  t = <v1, v2, …, vn> (n values)

- Each value is derived from an appropriate domain.

  v_i is a element of dom(Ai) or Null value

Exp: **<632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404) 894-2000">** is a tuple belonging to the CUSTOMER relation.

- A relation may be regarded as a ***set of tuples*** (rows).
- Columns in a table are also called attributes of the relation.

FORMAL DEFINITIONS

- A **domain** has a logical definition: e.g., "USA_phone_numbers" are the set of 10 digit phone numbers valid in the U.S.
- A domain may have a data-type or a format defined for it. The USA_phone_numbers may have a format: (ddd)-ddd-dddd where each d is a decimal digit. E.g., Dates have various formats such as monthname, date, year or yyyy-mm-dd, or dd mm,yyyy etc.
- An attribute designates the role played by the domain. E.g., the domain Date may be used to define attributes "Invoice-date" and "Payment-date".

FORMAL DEFINITIONS

- The relation is formed over the cartesian product of the sets; each set has values from a domain; that domain is used in a specific role which is conveyed by the attribute name.
- For example, attribute Cust-name is defined over the domain of strings of 25 characters. The role these strings play in the CUSTOMER relation is that of the name of customers.
- Formally,

Given $R(A_1, A_2, \ldots\ldots\ldots, A_n)$

$r(R) \subset \text{dom}(A_1) \times \text{dom}(A_2) \times \ldots \times \text{dom}(A_n)$

- R: schema of the relation
- r of R: a specific state or population of R.
- R is also called the **intension** of a relation
- r is also called the **extension** of a relation

FORMAL DEFINITIONS

- Let $S1 = \{0,1\}$
- Let $S2 = \{a,b,c\}$

- Let $r \subset S1 \times S2$

- Then for example: $r(R) = \{<0,a> , <0,b> , <1,c> \}$

  is one possible "state" or "population" or "extension" r of the relation R, defined over domains S1 and S2. It has three tuples.

DEFINITION SUMMARY

Informal Terms	Formal Terms
Table	Relation
Column	Attribute/Domain
Row	Tuple
Values in a column	Domain
Table Definition	Schema of a Relation
Populated Table	Extension

Example

Relation name

Attributes

Tuples

STUDENT	Name	SSN	HomePhone	Address	OfficePhone	Age	GPA
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21
	Katherine Ashly	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
	Charles Cooper	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

CHARACTERISTICS OF RELATIONS

- **Ordering of tuples in a relation r(R)**: The tuples are *not* considered to be ordered, even though 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 $R(A_1, A_2, ..., A_n)$ and the values in $t=<v_1, v_2, ..., v_n>$ to be *ordered* .
- **Values in a tuple**: All values are considered *atomic* (indivisible). A special **null** value is used to represent values that are unknown or inapplicable to certain tuple.

CHARACTERISTICS OF RELATIONS

- <u>Notation:</u>

- We refer to **component values** of a tuple t by $t[A_i] = v_i$ (the value of attribute A_i for tuple t).

  Similarly, $t[A_u, A_v, ..., A_w]$ refers to the subtuple of t containing the values of attributes $A_u, A_v, ..., A_w$, respectively.

CHARACTERISTICS OF RELATIONS

STUDENT	Name	SSN	HomePhone	Address	OfficePhone	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
	Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
	Katherine Ashly	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

t2[Name] = "Barbara Benson"
t2[Name, SSN] = "Barbara Benson", 533-69-1238

Relational Integrity Constraints (Schema –based contraints)

- Constraints are *conditions* that must hold on *all* valid relation instances. There are four main types of constraints:
  1. **Domain** constraints
  2. **Key** constraints
  3. **Entity integrity** constraints
  4. **Referential integrity** constraints

Key Constraints

- <u>**Superkey** of R:</u> A set of attributes SK of R such that no two tuples *in any valid relation instance r(R)* will have the same value for SK. That is, for any distinct tuples t1 and t2 in r(R), t1[SK] $\neq$ t2[SK].
- <u>**Key** of R:</u> A "minimal" superkey; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.

<u>**Example**</u>: The CAR relation schema:

CAR(<u>State</u>, <u>Reg#</u>, SerialNo, Make, Model, Year)

has two keys Key1 = {State, Reg#}, Key2 = {SerialNo}, which are also superkeys. {SerialNo, Make} is a superkey but *not* a key.

- If a relation has *several* **candidate keys**, one is chosen arbitrarily to be the **primary key**. The primary key attributes are *underlined*.

Key Constraints

The CAR relation with two candidate keys: LicenseNumber and EngineSerialNumber.

CAR	LicenseNumber	EngineSerialNumber	Make	Model	Year
	Texas ABC-739	A69352	Ford	Mustang	96
	Florida TVP-347	B43696	Oldsmobile	Cutlass	99
	New York MPO-22	X83554	Oldsmobile	Delta	95
	California 432-TFY	C43742	Mercedes	190-D	93
	California RSK-629	Y82935	Toyota	Camry	98
	Texas RSK-629	U028365	Jaguar	XJS	98

Entity Integrity Contraints

- **Relational Database Schema**: A set S of relation schemas that belong to the same database. S is the *name* of the **database**.

$$S = \{R_1, R_2, ..., R_n\}$$

- **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.

$$t[PK] \neq \text{null for any tuple } t \text{ in } r(R)$$

- Note: Other attributes of R may be similarly constrained to disallow null values, even though they are not members of the primary key.

Referential Integrity Constraints

- 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**.
- Tuples in the *referencing relation* R_1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the *referenced relation* R_2. A tuple t_1 in R_1 is said to **reference** a tuple t_2 in R_2 if $t_1[FK] = t_2[PK]$.
- Referential integrity constraints arise from the relationships among the entities
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from R_1.FK to R_2.

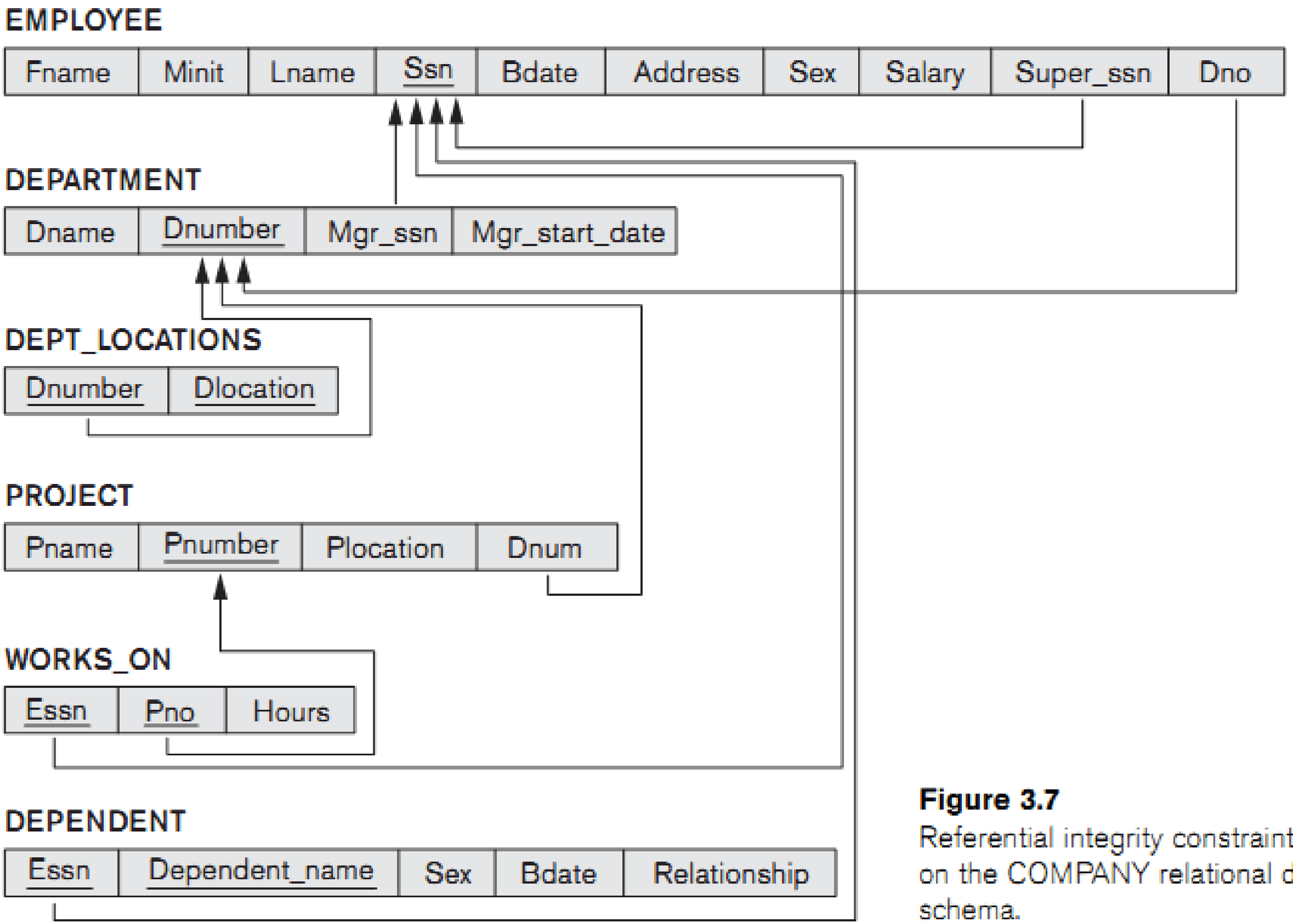


Figure 3.7
Referential integrity constraints displayed on the COMPANY relational database schema.

Referential Integrity Constraint

Statement of the constraint

The value in the foreign key column (or columns) FK of the the **referencing relation** R_1 can be either:

(1) a value of an existing primary key value of the corresponding primary key PK in the **referenced relation** R_2, or..

(2) a null.

In case (2), the FK in R_1 should not be a part of its own primary key.

Other Types of Constraints

Semantic Integrity Constraints:

- based on application semantics and cannot be expressed by the data model
- E.g., "the max. no. of hours per employee for all projects he or she works on is 56 hrs per week"
- SQL-99 allows triggers and ASSERTIONS to allow for some of these

Another type: Transition Contraints

PROJECT

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

EMPLOYEE

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

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

Case study

- Company Database

Example COMPANY Database

- Requirements of the Company (oversimplified for illustrative purposes)
  - The company is organized into DEPARTMENTs. Each department has a name, number and an employee who *manages* the department. We keep track of the start date of the department manager.
  - Each department *controls* a number of PROJECTs. Each project has a name, number and is located at a single location.

Example COMPANY Database

- We store each EMPLOYEE's social security number, address, salary, sex, and birthdate. Each employee *works for* one department but may *work on* several projects. We keep track of the number of hours per week that an employee currently works on each project. We also keep track of the *direct supervisor* of each employee.
- Each employee may *have* a number of DEPENDENTs. For each dependent, we keep track of their name, sex, birthdate, and relationship to employee.

Schema diagram for the COMPANY relational database schema; the primary keys are underlined.

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John		Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin		Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia		Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer		Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh		Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce		English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad		Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James		Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
		Houston
		Stafford
		Bellaire
		Sugarland

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

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

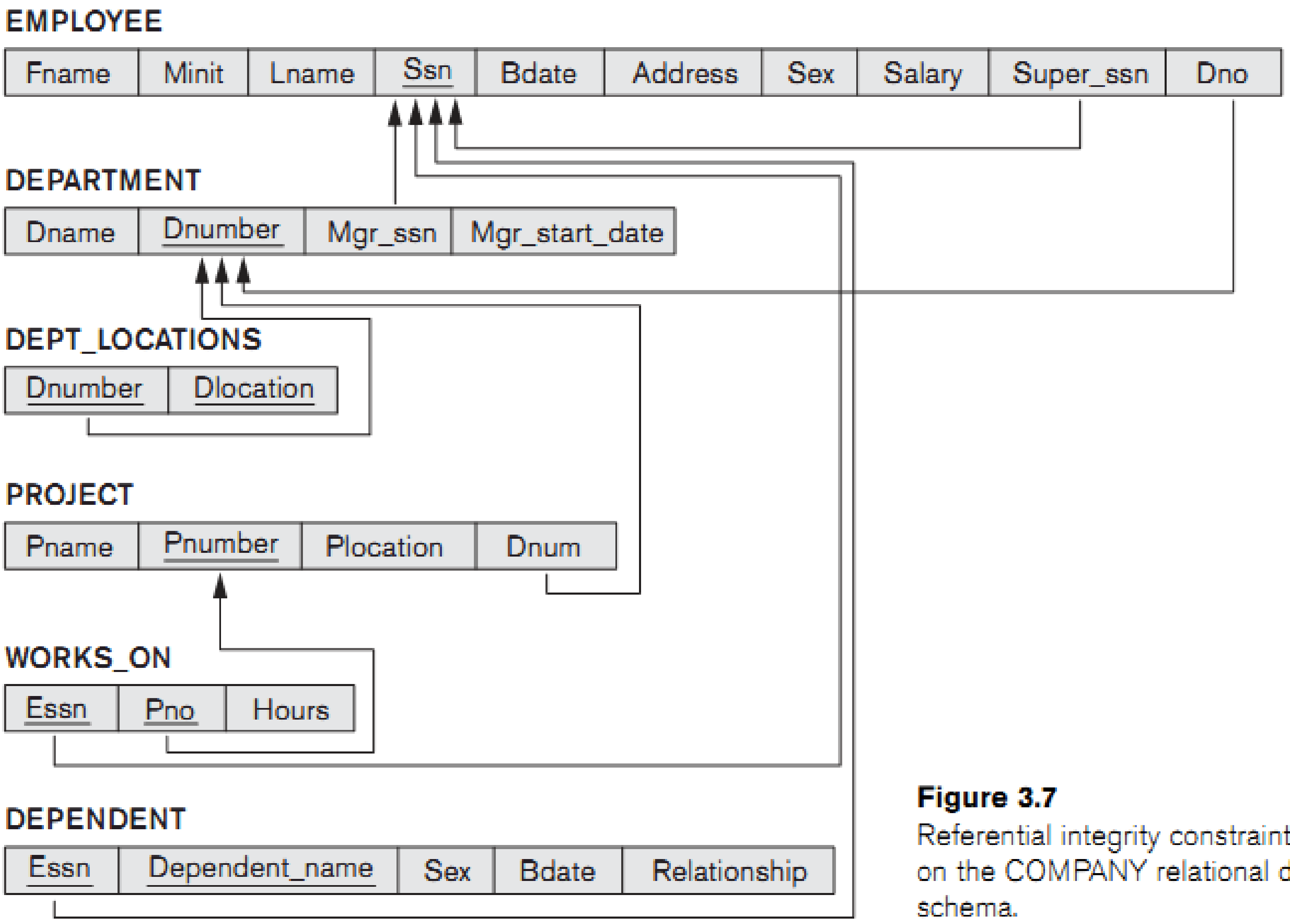


Figure 3.7
Referential integrity constraints displayed on the COMPANY relational database schema.

Update Operations on Relations

- INSERT a tuple.
- DELETE a tuple.
- MODIFY a tuple.

- Integrity constraints should not be violated by the update operations (valid state)
- 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.

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

- Examples:

Insert <'Cecilia', 'F', 'Kolonsky', NULL, '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, NULL, 4> into EMPLOYEE.

Insert <'Alicia', 'J', 'Zelaya', '999887777', '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, '987654321', 4> into EMPLOYEE.

Insert <'Cecilia', 'F', 'Kolonsky', '677678989', '1960-04-05', '6357 Windswept, Katy, TX', F, 28000, '987654321', 7> into EMPLOYEE.

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

Delete the EMPLOYEE tuple with Ssn = '999887777'.

Delete the EMPLOYEE tuple with Ssn = '333445555'.

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

Update the Dno of the EMPLOYEE tuple with Ssn = '999887777' to 1.

Update the Dno of the EMPLOYEE tuple with Ssn = '999887777' to 7.

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

Update Operations on Relations

- In case of integrity violation, several actions can be taken:
  - Cancel the operation that causes the violation (REJECT or RESTRICT 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

In-Class Exercise

(Taken from Exercise 5.15)

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

STUDENT(<u>SSN</u>, Name, Major, Bdate)

COURSE(<u>Course#</u>, Cname, Dept)

ENROLL(<u>SSN</u>, <u>Course#</u>, <u>Quarter</u>, Grade)

BOOK_ADOPTION(<u>Course#</u>, <u>Quarter</u>, Book_ISBN)

TEXT(<u>Book_ISBN</u>, Book_Title, Publisher, Author)

Draw a relational schema diagram specifying the foreign keys for this 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

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

Pname	Pnumber	Plocation	Dnu
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

f. Delete the WORKS_ON tuples with Essn = ‘333445555’.

g. Delete the EMPLOYEE tuple with Ssn = ‘987654321’.

h. Delete the PROJECT tuple with Pname = ‘ProductX’.

i. Modify the Mgr_ssn and Mgr_start_date of the DEPARTMENT tuple with Dnumber = 5 to ‘123456789’ and ‘2007-10-01’, respectively.

j. Modify the Super_ssn attribute of the EMPLOYEE tuple with Ssn = ‘999887777’ to ‘943775543’.

k. Modify the Hours attribute of the WORKS_ON tuple with Essn = ‘999887777’ and Pno = 10 to ‘5.0’.

Basic SQL

- SQL Data Definition & Data Types
- Specifying Constraints in SQL
- Basic Retrieval Queries in SQL
- INSERT, DELETE, UPDATE

SQL developments: an overview

- In 1986, ANSI and ISO published an initial standard for SQL: SQL-86 or SQL1
- In 1992, first major revision to ISO standard occurred, referred to as SQL2 or SQL-92
- In 1999, SQL-99 (SQL3) was released with support for object-oriented data management
- In late 2003, SQL-2003 was released
- Now: SQL-2006 was published

SQL

- *DDL: Create*, Alter, Drop
- *DML: Select*, *Insert*, *Update*, *Delete*
- DCL: Commit, Rollback, Grant, Revoke

Slide 2-81

CREATE SCHEMA

- Started with SQL 92
- A SQL Schema: is to group together tables and other constructs that belong to the same database application

CREATE SCHEMA SchemaName
  AUTHORIZATION AuthorizationIdentifier

```
CREATE SCHEMA COMPANY AUTHORIZATION 'Jsmith';
```

CREATE TABLE

- Specifies a new base relation by giving it a name, and specifying each of its attributes and their data types (INTEGER, FLOAT, DECIMAL(i,j), CHAR(n), VARCHAR(n))
- A constraint NOT NULL may be specified on an attribute

```
CREATE TABLE   DEPARTMENT
      (        DNAME           VARCHAR(10)  NOT NULL,
               DNUMBER         INTEGER      NOT NULL,
               MGRSSN          CHAR(9),
               MGRSTARTDATE          CHAR(9)  );
```

CREATE TABLE

- CREATE TABLE Company.TableName …

or

- CREATE TABLE TableName …

CREATE TABLE

CREATE TABLE TableName
({colName dataType [NOT NULL] [UNIQUE]
[DEFAULT defaultOption]
[CHECK searchCondition] **[,...]**}
[PRIMARY KEY (listOfColumns),]
{[UNIQUE (listOfColumns)] **[,…]**}
{[FOREIGN KEY (listOfFKColumns)
REFERENCES ParentTableName [(listOfCKColumns)]
[ON UPDATE referentialAction]
[ON DELETE referentialAction]] [,…]}
{[CHECK (searchCondition)] [,…] })

Data Types

- **Numeric:** INT or INTEGER, FLOAT or REAL, DOUBLE PRECISION, …
- **Character string:** fixed length CHAR(n), varying length VARCHAR(n)
- **Bit string:** BIT(n), e.g. B'1001'
- **Boolean:** true, false or NULL
- **DATE:** Made up of year-month-day in the format yyyy-mm-dd
- **TIME:** Made up of hour:minute:second in the format hh:mm:ss
- **TIME(i):** Made up of hour:minute:second plus i additional digits specifying fractions of a second format is hh:mm:ss:ii...i
- **TIMESTAMP:** Has both DATE and TIME components

Data Types

- A domain can be declared and used with the attribute specification

CREATE DOMAIN DomainName AS DataType [CHECK conditions];

Example:

```
CREATE DOMAIN SSN_TYPE AS CHAR(9);
```

Specifying Constraints in SQL

- Specifying Attribute Constraints and Attribute Defaults
- Default values
  - DEFAULT <value> can be specified for an attribute
  - If no default clause is specified, the default value is NULL for attributes that do not have the NOT NULL constraint
- CHECK clause: restrict attribute or domain values

  DNUMBER INT NOT NULL CHECK (DNUMBER>0 AND DNUMBER<21);

  - CREATE DOMAIN can also be used in conjunction with the CHECK clause:

  CREATE DOMAIN D_NUM AS INTEGER CHECK (D_NUM>0 AND D_NUM<21);

Specifying Constraints in SQL

- Specifying Key Constraints
- Key attributes can be specified via the PRIMARY KEY and UNIQUE phrases

```
CREATE TABLE   DEPT
    (   DNAME           VARCHAR(10)  NOT NULL,
        DNUMBER         INTEGER      NOT NULL,
        MGRSSN          CHAR(9),
        MGRSTARTDATE         CHAR(9),
        PRIMARY KEY (DNUMBER),
        UNIQUE (DNAME),
        FOREIGN KEY (MGRSSN) REFERENCES EMP  );
Or      Dnumber  INTEGER PRIMARY KEY;
```

REFERENTIAL INTEGRITY OPTIONS

- Specifying Referential Integrity Constraints: FOREIGN KEY clause. Can specify RESTRICT, CASCADE, SET NULL or SET DEFAULT on referential integrity constraints

```
CREATE TABLE   DEPT
 (    DNAME         VARCHAR(10)        NOT NULL,
      DNUMBER   INTEGER      NOT NULL,
      MGRSSN       CHAR(9),
      MGRSTARTDATE  CHAR(9),
      PRIMARY KEY (DNUMBER),
      UNIQUE (DNAME),
      FOREIGN KEY (MGRSSN) REFERENCES EMP
ON DELETE SET DEFAULT ON UPDATE CASCADE
);
```

Specifying Constraints in SQL

- Giving names to constraints

```
CREATE TABLE EMPLOYEE
    (   ...,
        Dno          INT                NOT NULL       DEFAULT 1,
    CONSTRAINT EMPPK
        PRIMARY KEY (Ssn),
    CONSTRAINT EMPSUPERFK
        FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
                    ON DELETE SET NULL          ON UPDATE CASCADE,
    CONSTRAINT EMPDEPTFK
        FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber)
                    ON DELETE SET DEFAULT       ON UPDATE CASCADE);
```

Specifying Constraints in SQL

- Specifying Constraints on Tuples (tuple-based) using CHECK: at the end of CREATE TABLE
- Example:

```
CHECK (Dept_create_date <= Mgr_start_date);
```

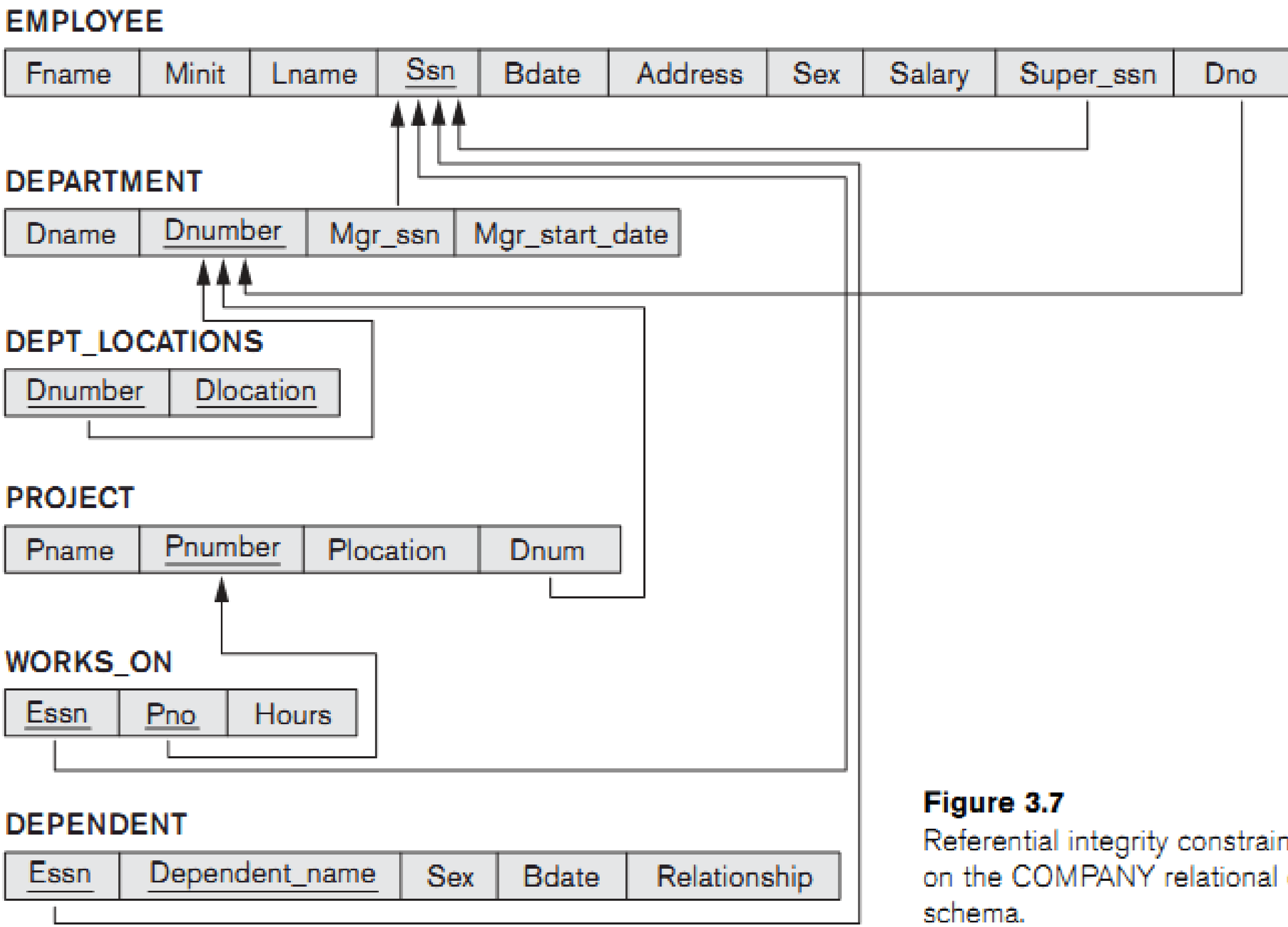


Figure 3.7
Referential integrity constraints displayed on the COMPANY relational database schema.

CREATE TABLE

```
CREATE TABLE TableName
({colName dataType [NOT NULL] [UNIQUE]
[DEFAULT defaultOption]
[CHECK searchCondition] [,...]}
[PRIMARY KEY (listOfColumns),]
{[UNIQUE (listOfColumns)] [,…]}
{[FOREIGN KEY (listOfFKColumns)
 REFERENCES ParentTableName [(listOfCKColumns)]
 [ON UPDATE referentialAction]
 [ON DELETE referentialAction ]] [,…]}
{[CHECK (searchCondition)] [,…] })
```

```
CREATE TABLE EMPLOYEE
        ( Fname             VARCHAR(15)       NOT NULL,
          Minit             CHAR,
          Lname             VARCHAR(15)       NOT NULL,
          Ssn               CHAR(9)           NOT NULL,
          Bdate             DATE,
          Address           VARCHAR(30),
          Sex               CHAR,
          Salary            DECIMAL(10,2),
          Super_ssn         CHAR(9),
          Dno               INT               NOT NULL,
        PRIMARY KEY (Ssn),
        FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE DEPARTMENT
        ( Dname             VARCHAR(15)       NOT NULL,
          Dnumber           INT               NOT NULL,
          Mgr_ssn           CHAR(9)           NOT NULL,
          Mgr_start_date    DATE,
        PRIMARY KEY (Dnumber),
        UNIQUE (Dname),
        FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
CREATE TABLE DEPT_LOCATIONS
        ( Dnumber           INT               NOT NULL,
          Dlocation         VARCHAR(15)       NOT NULL,
        PRIMARY KEY (Dnumber, Dlocation),
        FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );
```

Basic Retrieval Queries in SQL

- SELECT statement
- SQL relation (table) is a *multi-set* (sometimes called a bag) of tuples; it *is not* a set of tuples
- SQL relations can be constrained to be sets by specifying PRIMARY KEY or UNIQUE attributes, or by using the DISTINCT option in a query

Basic Retrieval Queries in SQL (cont.)

- Basic form of the SQL SELECT statement is called a *mapping* or a *SELECT-FROM-WHERE block*

SELECT <attribute list>
FROM <table list>
WHERE <condition>

  - <attribute list> is a list of attribute names whose values are to be retrieved by the query
  - <table list> is a list of the relation names required to process the query
  - <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query

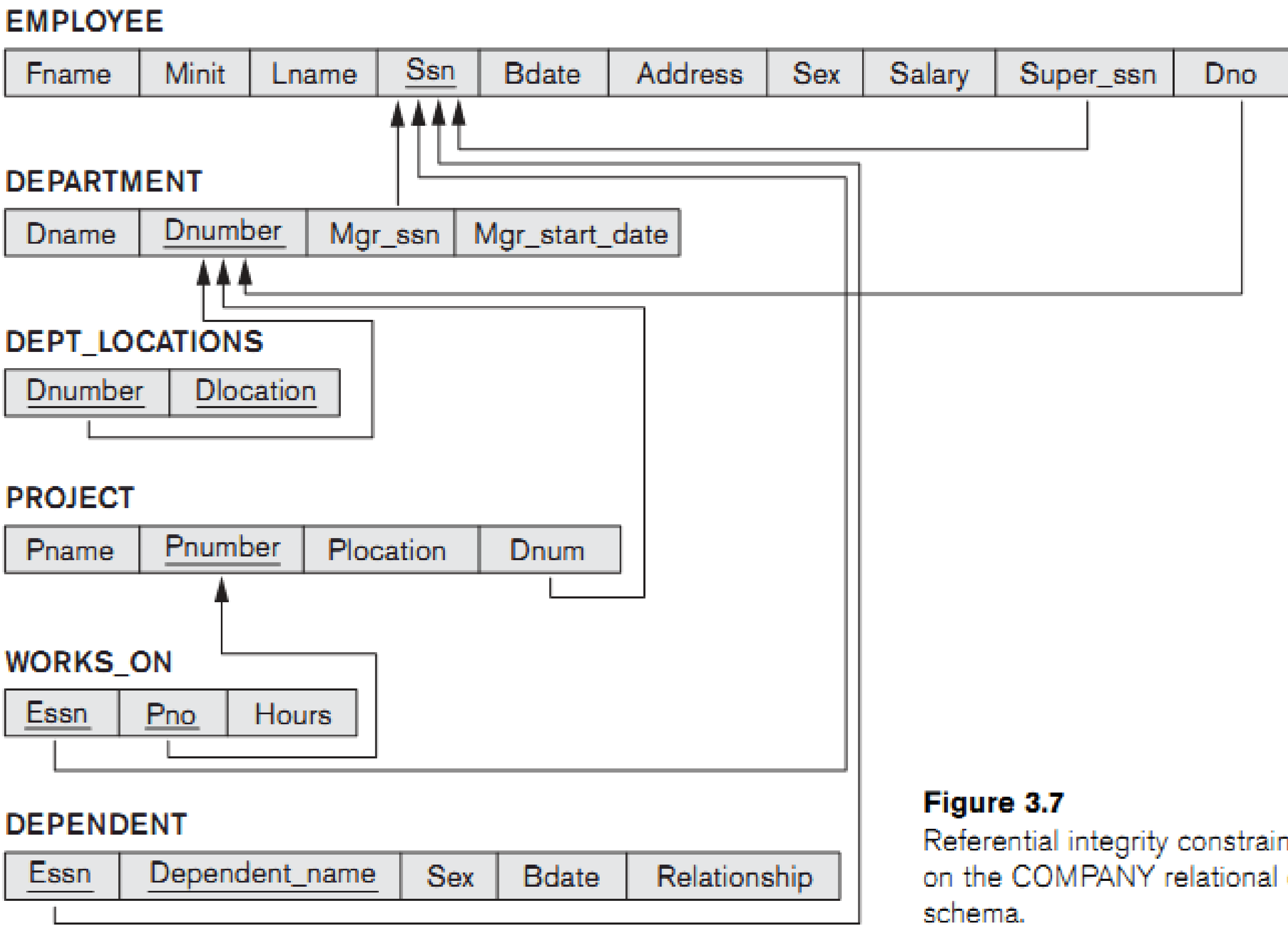


Figure 3.7
Referential integrity constraints displayed on the COMPANY relational database schema.

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

Simple SQL Queries

- All subsequent examples use the COMPANY database
- Example of a simple query on *one* relation
- Query 0: Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.

```
Q0: SELECT    BDATE, ADDRESS
    FROM      EMPLOYEE
    WHERE     FNAME='John' AND MINIT='B’
    AND       LNAME='Smith’
```

  – The SELECT-clause specifies the projection attributes and the WHERE-clause specifies the selection condition
  – The result of the query may contain duplicate tuples

Simple SQL Queries (cont.)

- Query 1: Retrieve the name and address of all employees who work for the 'Research' department.

```
Q1: SELECT   FNAME, LNAME, ADDRESS
    FROM     EMPLOYEE, DEPARTMENT
    WHERE    DNAME='Research' AND
  DNUMBER=DNO
```

  – (DNAME='Research') is a ***selection condition***
  – (DNUMBER=DNO) is a ***join condition***

Simple SQL Queries (cont.)

- Query 2: For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate.

```
Q2: SELECT   PNUMBER, DNUM, LNAME, BDATE, ADDRESS
    FROM     PROJECT, DEPARTMENT, EMPLOYEE
    WHERE    DNUM=DNUMBER AND MGRSSN=SSN
    AND      PLOCATION='Stafford'
```

  - In Q2, there are *two* join conditions
  - The join condition DNUM=DNUMBER relates a project to its controlling department
  - The join condition MGRSSN=SSN relates the controlling department to the employee who manages that department

Aliases, * and DISTINCT, Empty WHERE-clause

- In SQL, we can use the same name for two (or more) attributes as long as the attributes are in *different relations* A query that refers to two or more attributes with the same name must *qualify* the attribute name with the relation name by *prefixing* the relation name to the attribute name

<u>Example:</u>

- EMPLOYEE.LNAME, DEPARTMENT.DNAME

ALIASES

- Some queries need to refer to the same relation twice
- In this case, *aliases* are given to the relation name
- Query 8: For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

Q8: SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM EMPLOYEE E S
WHERE E.SUPERSSN=S.SSN

  – In Q8, the alternate relation names E and S are called *aliases* or *tuple variables* for the EMPLOYEE relation
  – We can think of E and S as two *different copies* of EMPLOYEE; E represents employees in role of *supervisees* and S represents employees in role of *supervisors*

ALIASES (cont.)

- Aliasing can also be used in any SQL query for convenience
  Can also use the AS keyword to specify aliases

```
Q8: SELECT   E.FNAME, E.LNAME, S.FNAME,
             S.LNAME
    FROM     EMPLOYEE AS E, EMPLOYEE AS S
    WHERE    E.SUPERSSN=S.SSN
```

UNSPECIFIED WHERE-clause

- A *missing WHERE-clause* indicates no condition; hence, *all tuples* of the relations in the FROM-clause are selected
- This is equivalent to the condition WHERE TRUE
- Query 9: Retrieve the SSN values for all employees.

```
Q9:     SELECT  SSN
        FROM    EMPLOYEE
```

- If more than one relation is specified in the FROM-clause *and* there is no join condition, then the *CARTESIAN PRODUCT* of tuples is selected

UNSPECIFIED WHERE-clause (cont.)

- Example:

```
Q10:    SELECT  SSN, DNAME
        FROM    EMPLOYEE, DEPARTMENT
```

  – It is extremely important not to overlook specifying any selection and join conditions in the WHERE-clause; otherwise, incorrect and very large relations may result

USE OF *

- To retrieve all the attribute values of the selected tuples, a * is used, which stands for *all the attributes*

  Examples:

Q1C:	**SELECT**	*****
	FROM	**EMPLOYEE**
	WHERE	**DNO=5**
Q1D:	**SELECT**	*****
	FROM	**EMPLOYEE, DEPARTMENT**
	WHERE	**DNAME='Research' AND DNO=DNUMBER**

USE OF DISTINCT

- SQL does not treat a relation as a set; *duplicate tuples can appear*
- To eliminate duplicate tuples in a query result, the keyword **DISTINCT** is used
- For example, the result of Q11 may have duplicate SALARY values whereas Q11A does not have any duplicate values

```
Q11:   SELECT   SALARY
       FROM     EMPLOYEE
Q11A:  SELECT   DISTINCT SALARY
       FROM     EMPLOYEE
```

SUBSTRING COMPARISON

- The **LIKE** comparison operator is used to compare partial strings
- '%' (or '*' in some implementations) replaces an arbitrary number of characters
- '_' replaces a single arbitrary character

SUBSTRING COMPARISON (cont.)

- Query 25: Retrieve all employees whose address is in Houston, Texas. Here, the value of the ADDRESS attribute must contain the substring 'Houston,TX'.

Q25: **SELECT** **FNAME, LNAME**
FROM **EMPLOYEE**
WHERE **ADDRESS LIKE '%Houston,TX%'**

SUBSTRING COMPARISON (cont.)

- Query 26: Retrieve all employees who were born during the 1950s. Here, '5' must be the 8th character of the string (according to our format for date), so the BDATE value is '_______5_', with each underscore as a place holder for a single arbitrary character.

```
Q26:    SELECT   FNAME, LNAME
        FROM     EMPLOYEE
        WHERE    BDATE LIKE    '_______5_'
```

- The LIKE operator allows us to get around the fact that each value is considered atomic and indivisible; hence, in SQL, character string attribute values are not atomic

ARITHMETIC OPERATIONS

- The standard arithmetic operators '+', '-'. '*', and '/‘ can be applied to numeric values in an SQL query result
- Query 27: Show the effect of giving all employees who work on the 'ProductX' project a 10% raise.

```
Q27: SELECT    FNAME, LNAME, 1.1*SALARY
               FROM  EMPLOYEE, WORKS_ON, PROJECT
     WHERE     SSN=ESSN AND PNO=PNUMBER AND
               PNAME='ProductX’
```

Specifying Updates in SQL

- There are three SQL commands to modify the database; INSERT, DELETE, and UPDATE

INSERT

- To add one or more tuples to a relation
- Attribute values should be listed in the same order as the attributes were specified in the CREATE TABLE command

INSERT (cont.)

- <u>Example:</u>

  U1: INSERT INTO EMPLOYEE
  VALUES ('Richard','K','Marini', '653298653', '30-DEC-52', '98 Oak Forest,Katy,TX', 'M', 37000,'987654321', 4)

- An alternate form of INSERT specifies explicitly the attribute names that correspond to the values in the new tuple
- Attributes with NULL values can be left out
- <u>Example:</u> Insert a tuple for a new EMPLOYEE for whom we only know the FNAME, LNAME, and SSN attributes.

  U1A: INSERT INTO EMPLOYEE (FNAME, LNAME, SSN)
  VALUES ('Richard', 'Marini', '653298653')

INSERT (cont.)

- Important Note: Only the constraints specified in the DDL commands are automatically enforced by the DBMS when updates are applied to the database
- Another variation of INSERT allows insertion of *multiple tuples* resulting from a query into a relation

INSERT (cont.)

- Example: Suppose we want to create a temporary table that has the name, number of employees, and total salaries for each department. A table DEPTS_INFO is created by U3A, and is loaded with the summary information retrieved from the database by the query in U3B.

```
U3A:    CREATE TABLE DEPTS_INFO
        (DEPT_NAME      VARCHAR(10),
        NO_OF_EMPS      INTEGER,
        TOTAL_SAL  INTEGER);

U3B:    INSERT INTO DEPTS_INFO (DEPT_NAME,
                    NO_OF_EMPS, TOTAL_SAL)
        SELECT DNAME, COUNT (*), SUM
(SALARY)
        FROM        DEPARTMENT, EMPLOYEE
        WHERE       DNUMBER=DNO
        GROUP BY    DNAME ;
```

INSERT (cont.)

- Note: The DEPTS_INFO table may not be up-to-date if we change the tuples in either the DEPARTMENT or the EMPLOYEE relations *after* issuing U3B. We have to create a view (see later) to keep such a table up to date.

DELETE

- Removes tuples from a relation
- Includes a WHERE-clause to select the tuples to be deleted
- Tuples are deleted from only *one table* at a time (unless CASCADE is specified on a referential integrity constraint)
- A missing WHERE-clause specifies that *all tuples* in the relation are to be deleted; the table then becomes an empty table
- The number of tuples deleted depends on the number of tuples in the relation that satisfy the WHERE-clause
- Referential integrity should be enforced

DELETE (cont.)

- Examples:

```
U4A:    DELETE FROM   EMPLOYEE
        WHERE         LNAME='Brown'

U4B:    DELETE FROM   EMPLOYEE
        WHERE         SSN='123456789'

U4C:    DELETE FROM   EMPLOYEE
        WHERE         DNO IN
               (SELECT  DNUMBER
                FROM    DEPARTMENT
                WHERE   DNAME='Research')

U4D:    DELETE FROM   EMPLOYEE
```

UPDATE

- Used to modify attribute values of one or more selected tuples
- A WHERE-clause selects the tuples to be modified
- An additional SET-clause specifies the attributes to be modified and their new values
- Each command modifies tuples *in the same relation*
- Referential integrity should be enforced

UPDATE (cont.)

- Example: Change the location and controlling department number of project number 10 to 'Bellaire' and 5, respectively.

U5: UPDATE PROJECT
SET PLOCATION = 'Bellaire', DNUM = 5
WHERE PNUMBER=10

UPDATE (cont.)

- Example: Give all employees in the 'Research' department a 10% raise in salary.

```
U6:  UPDATE    EMPLOYEE
     SET       SALARY = SALARY *1.1
     WHERE     DNO IN (SELECT   DNUMBER
                       FROM     DEPARTMENT
                       WHERE    DNAME='Research')
```

- In this request, the modified SALARY value depends on the original SALARY value in each tuple
- The reference to the SALARY attribute on the right of = refers to the old SALARY value before modification
- The reference to the SALARY attribute on the left of = refers to the new SALARY value after modification

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

Summary of SQL Queries

- A query in SQL can consist of up to six clauses, but only the first two, SELECT and FROM, are mandatory. The clauses are specified in the following order:

SELECT <attribute list>
FROM <table list>
[WHERE <condition>]
[GROUP BY <grouping attribute(s)>]
[HAVING <group condition>]
[ORDER BY <attribute list>]