An Overview Lecture on

Database Systems

Olaf Hartig

olaf.hartig@liu.se

LINKÖPING UNIVERSITY

Data Management before Database Systems

- Traditional **file processing**: each user defines and implements the files needed for a specific software application

- Once both the user base and the application base grow
  - many shared files
  - a multitude of file structures
  - a need to exchange data among applications

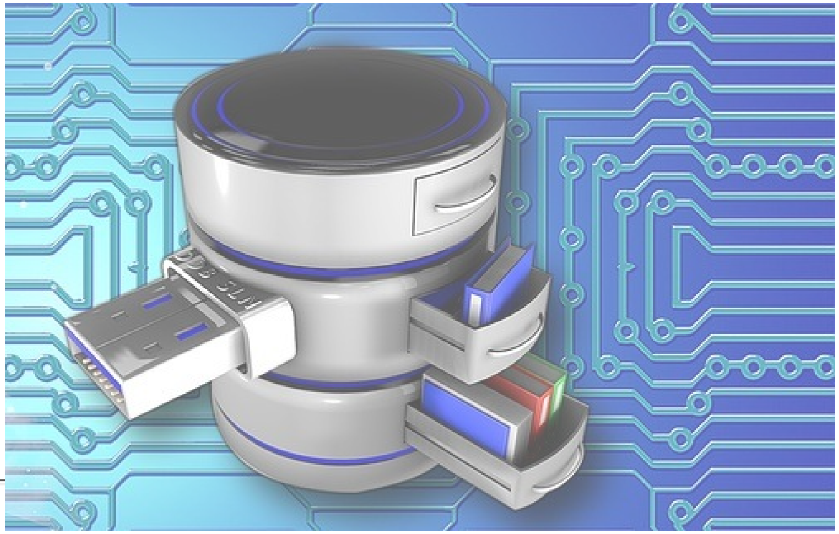
- Problems
  - multiple copies (redundancy)
  - independent updates (inconsistency)
  - concurrent updates (inaccuracy)
  - multiple formats (incompatibility)
  - proliferation (insecurity)
  - poor chain of responsibility (inauditability)
  - changes are difficult to apply (inflexibility)

https://cdn.pixabay.com/photo/2014/06/01/22/26/clutter-360058_960_720.jpg

Database Approach

- Recognition that data is a critical corporate asset (along with capital and personnel)
  - need to manage the data in a more systematic manner

- Database approach: Use a *single repository* to maintain data that is defined once and accessed by various users
  - addresses the aforementioned problems



https://cdn.pixabay.com/photo/2017/06/12/04/21/database-2394312_960_720.jpg

Basic Terminology and Characteristics of Database Systems

Most Basic Terminology

- **Data**: known facts that can be recorded and that have implicit meaning

- **Database**: logically coherent collection of related data
  - Built for a specific purpose
  - Represents some aspects of the real world (**miniworld**)

- Examples of large databases
  - Amazon.com's product data
  - Data collection underlying Webreg

Example of a Database

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

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Example from “Fundamentals of Database Systems” by Elmasri and Navathe, Addison Wesley.

Terminology (cont'd)

- **Database management system** (**DBMS**)
  - Collection of computer programs
  - Enables users to create and maintain a DB
  - Supports concurrent access to the DB by multiple users and programs
  - Protects the DB against unauthorized access and manipulation
  - Provides means to evolve the DB as requirements change

- Examples of database management systems
  - IBM’s DB2, Microsoft’s Access, Microsoft’s SQL Server, Oracle, SAP’s SQL Anywhere, MySQL, PostgreSQL

Database System

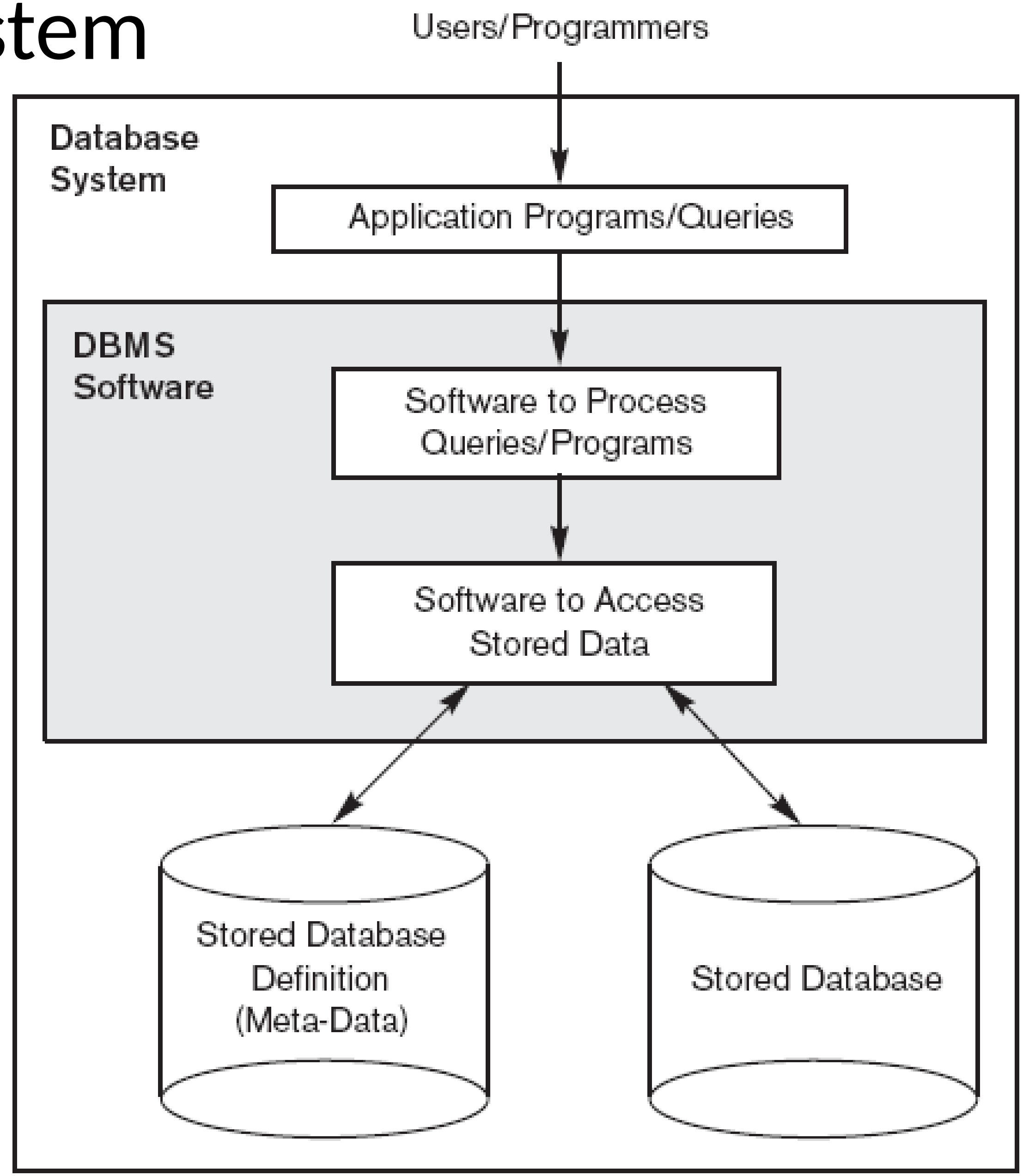


Figure from “Fundamentals of Database Systems” by Elmasri and Navathe, Addison Wesley.

Main Characteristics of Database Systems

- Application programs isolated from data through **abstraction**
  - DBMS does not expose details of how (or where) data is stored or how operations are implemented
  - Instead, programs operate on an abstract model of the data, rather than referring to data storage details
  - Data structures and storage organization can be changed without having to change the application programs

- Support of multiple **views** of the data
  - Different users may see different views of the database, which contain only the data of interest to these users

Main Characteristics of Database Systems

- **Self-describing**
  - DBS contains a *database catalog* with meta-data that describes structure and constraints of the database(s)
  - Database catalog used by DBMS, and by DB users who need information about DB structure
  - Example:

RELATIONS

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

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

Example from “Fundamentals of Database Systems” by Elmasri and Navathe, Addison Wesley.

Database System Design Process

- Two main activities:
  - **Database design** focuses on defining the database
  - **Application design** focuses on the programs and interfaces that access the database (out of scope of this lecture)

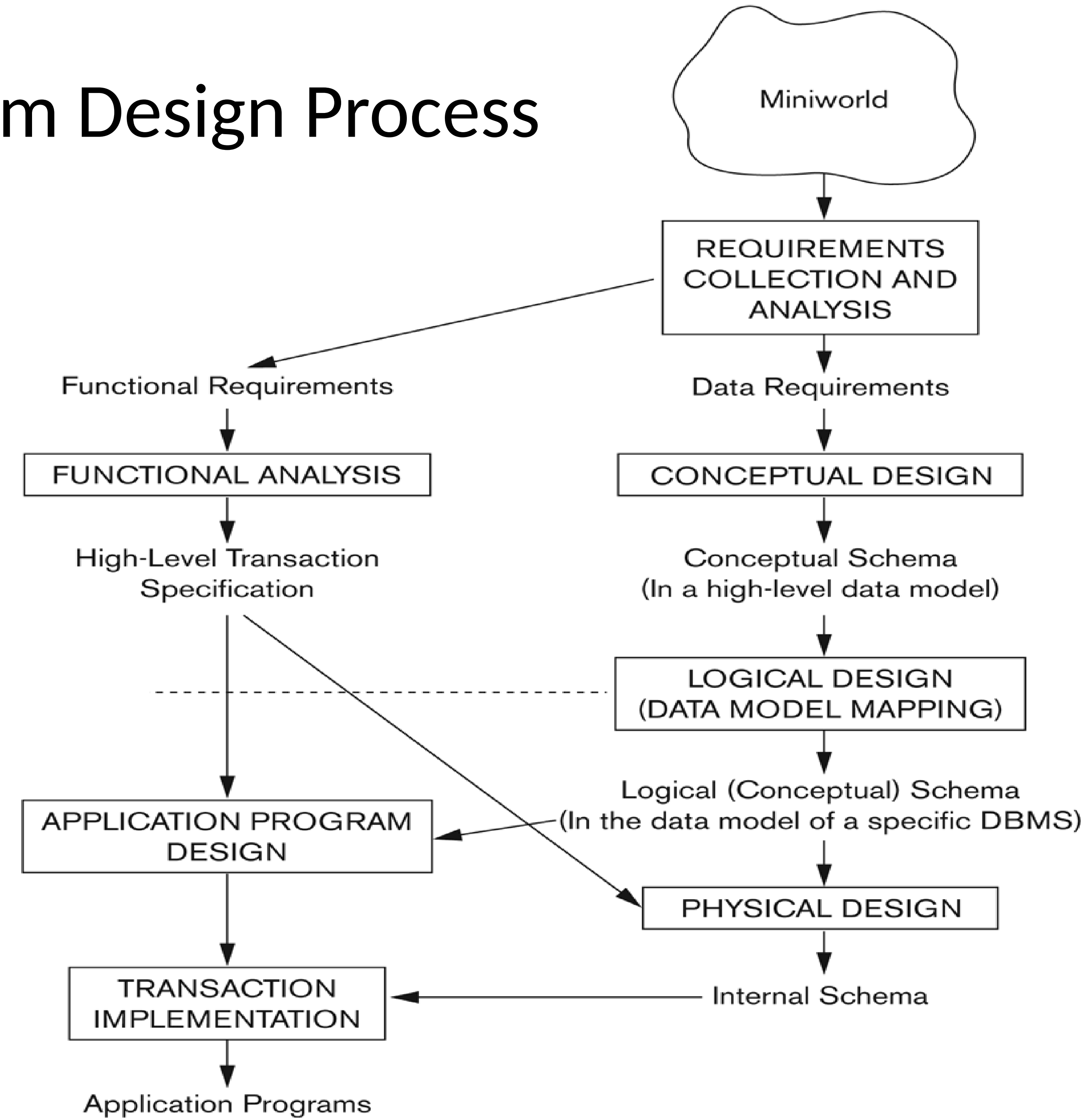


Figure from “Fundamentals of Database Systems” by Elmasri and Navathe, Addison Wesley.

Conceptual Design

using the Entity-Relationship (ER) Model

Conceptual Design in the DBS Design Process

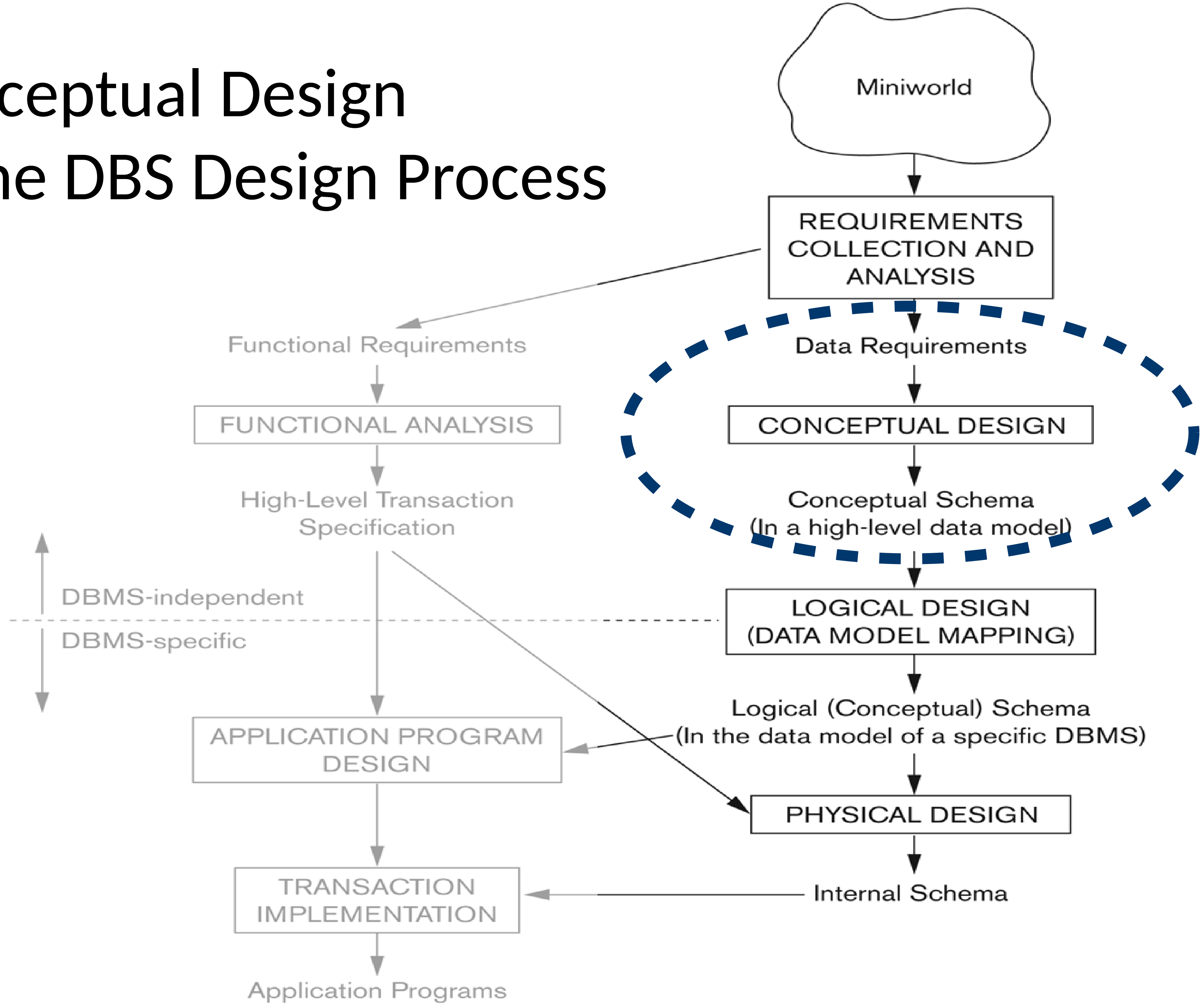


Figure from “Fundamentals of Database Systems” by Elmasri and Navathe, Addison Wesley.

Example of Data Requirements

A taxi company needs to model their activities.

There are two types of employees in the company: drivers and operators. For drivers it is interesting to know the date of issue and type of the driving license, and the date of issue of the taxi driver's certificate. For all employees it is interesting to know their personal number, address and the available phone numbers.

The company owns a number of cars. For each car there is a need to know its type, year of manufacturing, number of places in the car and date of the last service.

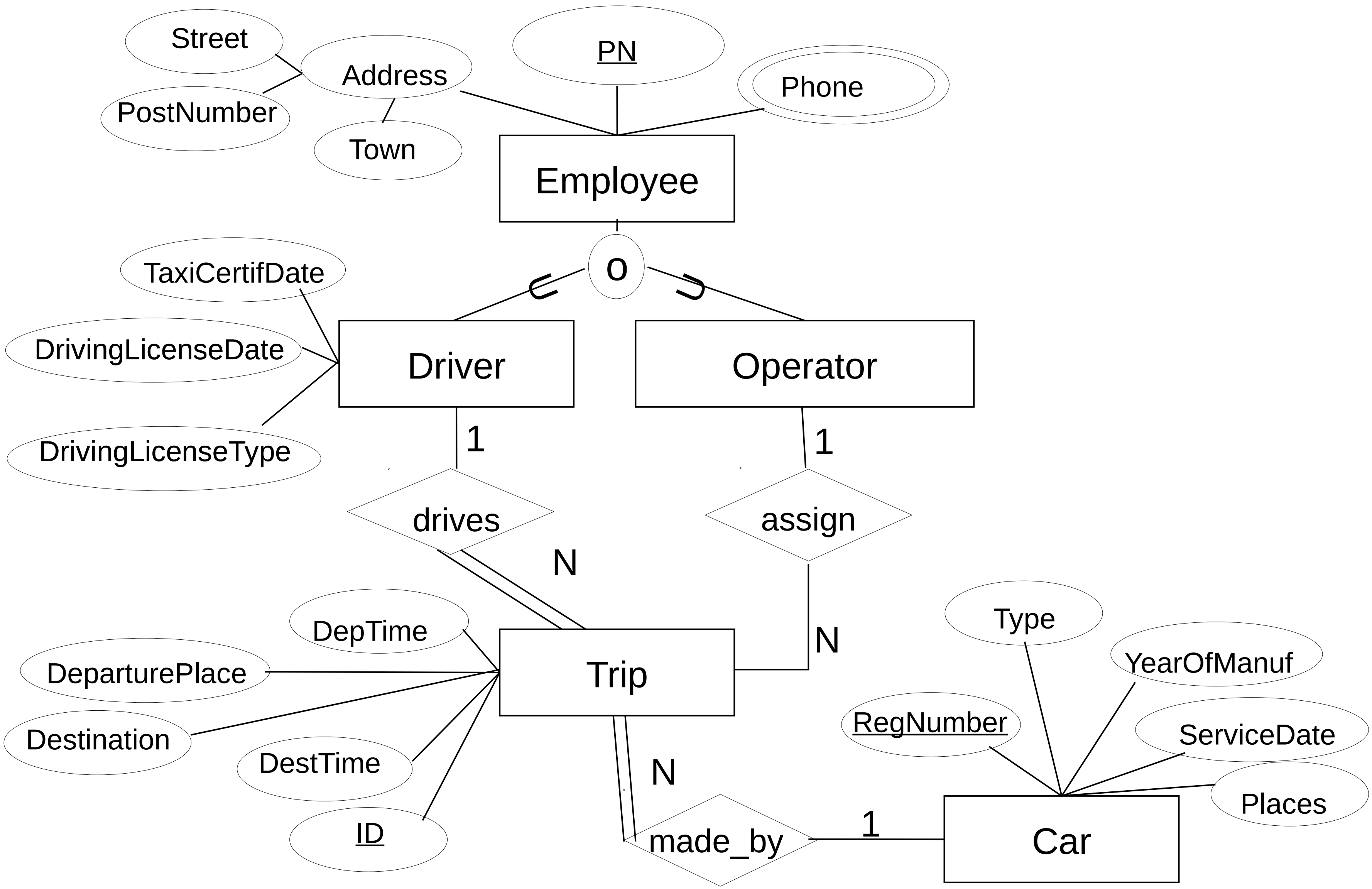
The company wants to have a record of car trips. A taxi may be picked on a street or ordered through an operator who assigns the order to a certain driver and a car. Departure and destination addresses together with times should also be recorded.

Entity-Relationship (ER) Model

- High-level conceptual data model
  - An overview of the database
  - Easy to translate to data model of DBMS
  - Easy to discuss with non-database experts

- ER diagram
  - Diagrammatic notation associated with the ER model

Example Entity-Relationship Diagram



Logical Design

using the Relational Data Model

Logical Design
in the DBS Design Process

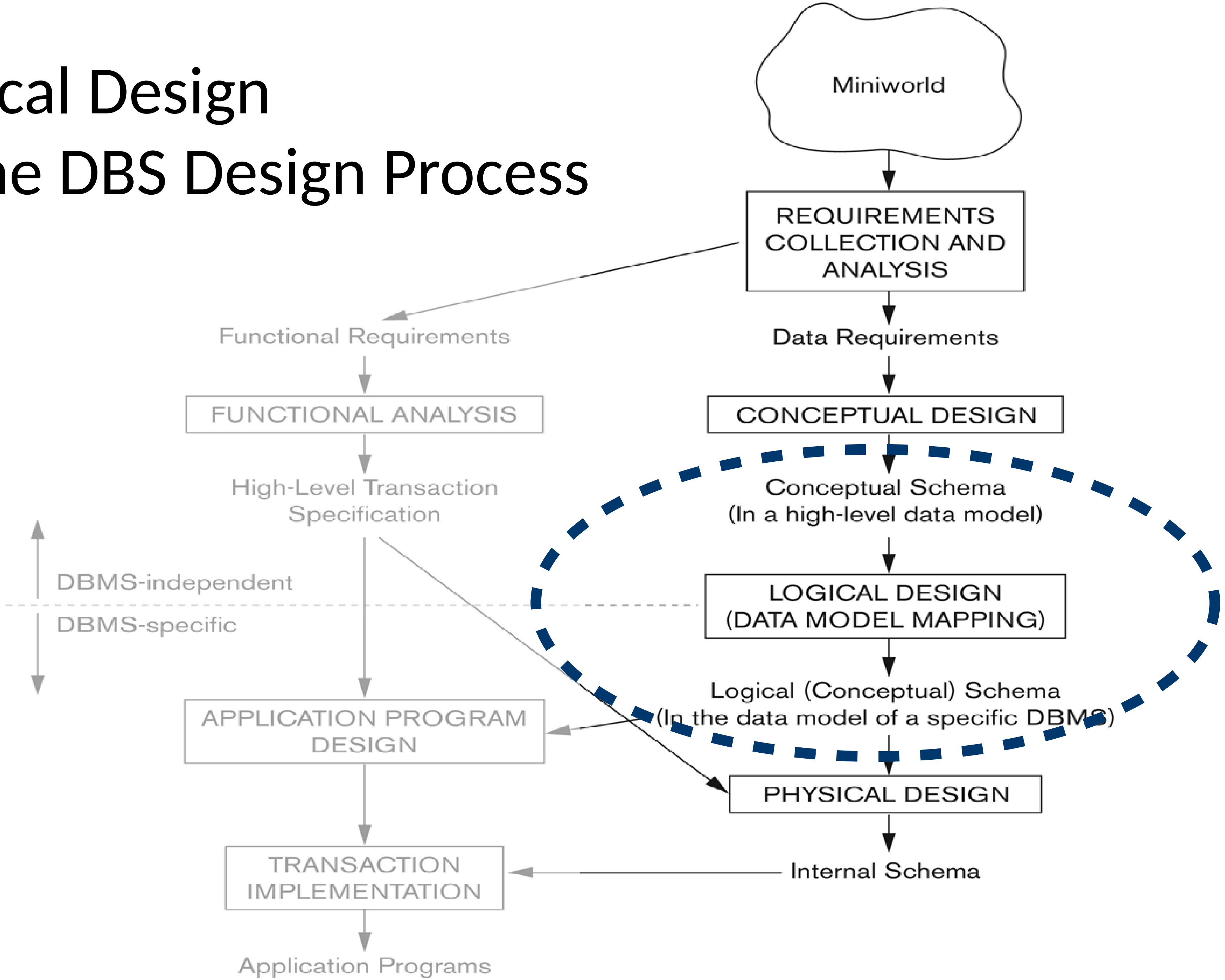


Figure from “Fundamentals of Database Systems” by Elmasri and Navathe, Addison Wesley.

Relational Model

- Relational database: represent data as a collection of *relations*
  - Think of a relation as a table of values

Relation Name

Attributes

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Tuples

- Each row (*tuple*) represents a record of related data values
  - Facts that typically correspond to a real-world entity or relationship
- Each column (*attribute*) holds a corresponding value for each row
  - Columns associated with a data type (*domain*)
  - Each column header: *attribute name*

Example from “Fundamentals of Database Systems” by Elmasri and Navathe, Addison Wesley.

Relational Model (cont'd)

- Relational database: represent data as a collection of *relations*
  - Think of a relation as a table of values

Relation Name

Attributes

STUDENT

Tuples

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

- **Schema** describes the relation
  - Relation name, attribute names, and attribute types
  - Integrity constraints
- **Instance** denotes the *current* contents of the relation (also called *state*)
  - Set of tuples

Example from “Fundamentals of Database Systems” by Elmasri and Navathe, Addison Wesley.

Integrity Constraints

- **Constraints** are restrictions on the permitted values in a DB state
  - Derived from the rules in the miniworld that the DB represents

Relation Name

Attributes

STUDENT

Tuples

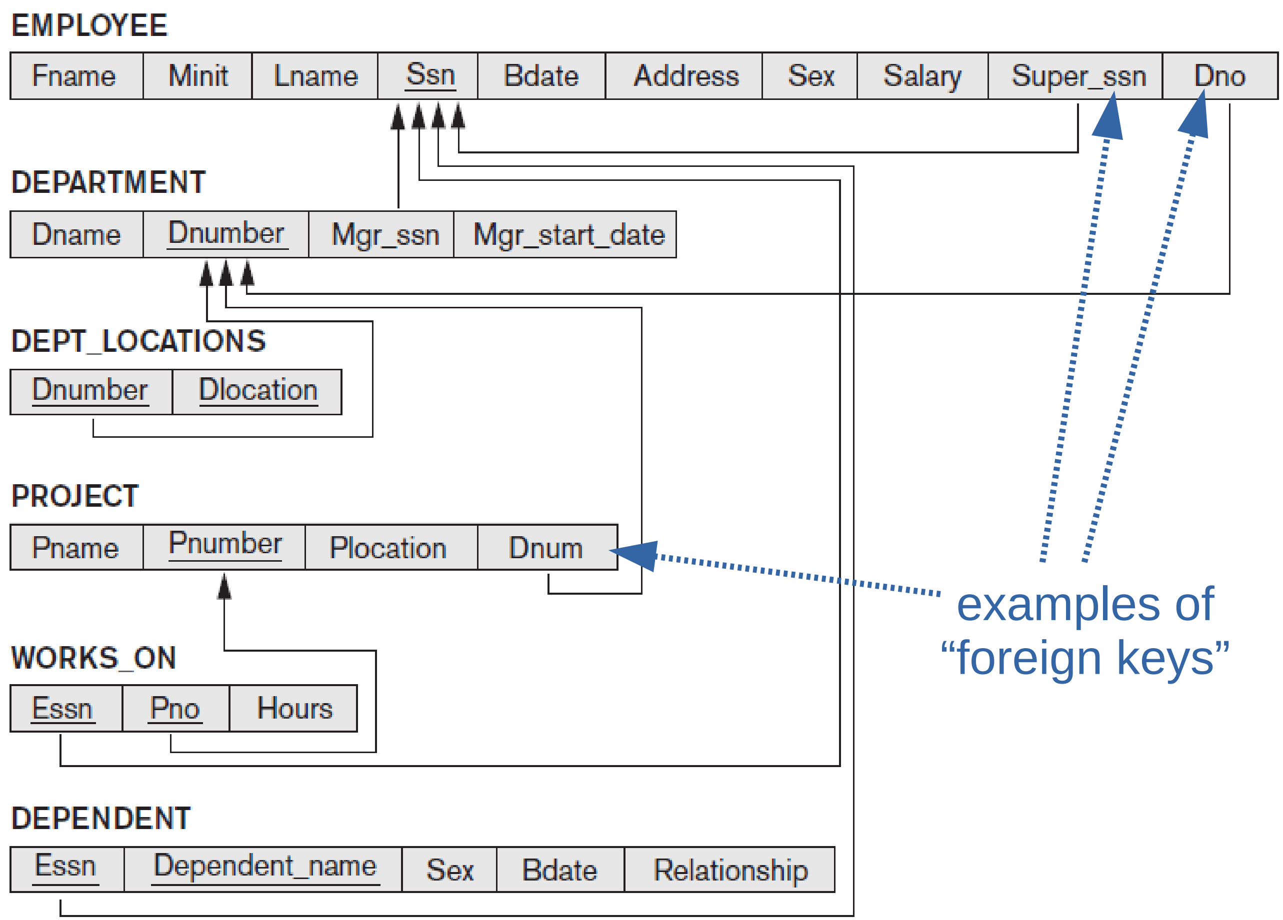
Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

- Examples:
  - Uniqueness constraint
  - Entity integrity constraint (no NULL value)

Example from “Fundamentals of Database Systems” by Elmasri and Navathe, Addison Wesley.

Referential Integrity Constraints

- Specified between two relations
- Allows tuples in one relation to *refer to* tuples in another
- Maintains consistency among tuples in two relations



Example from "Fundamentals of Database Systems" by Elmasri and Navathe, Addison Wesley.

Logical Design
in the DBS Design Process

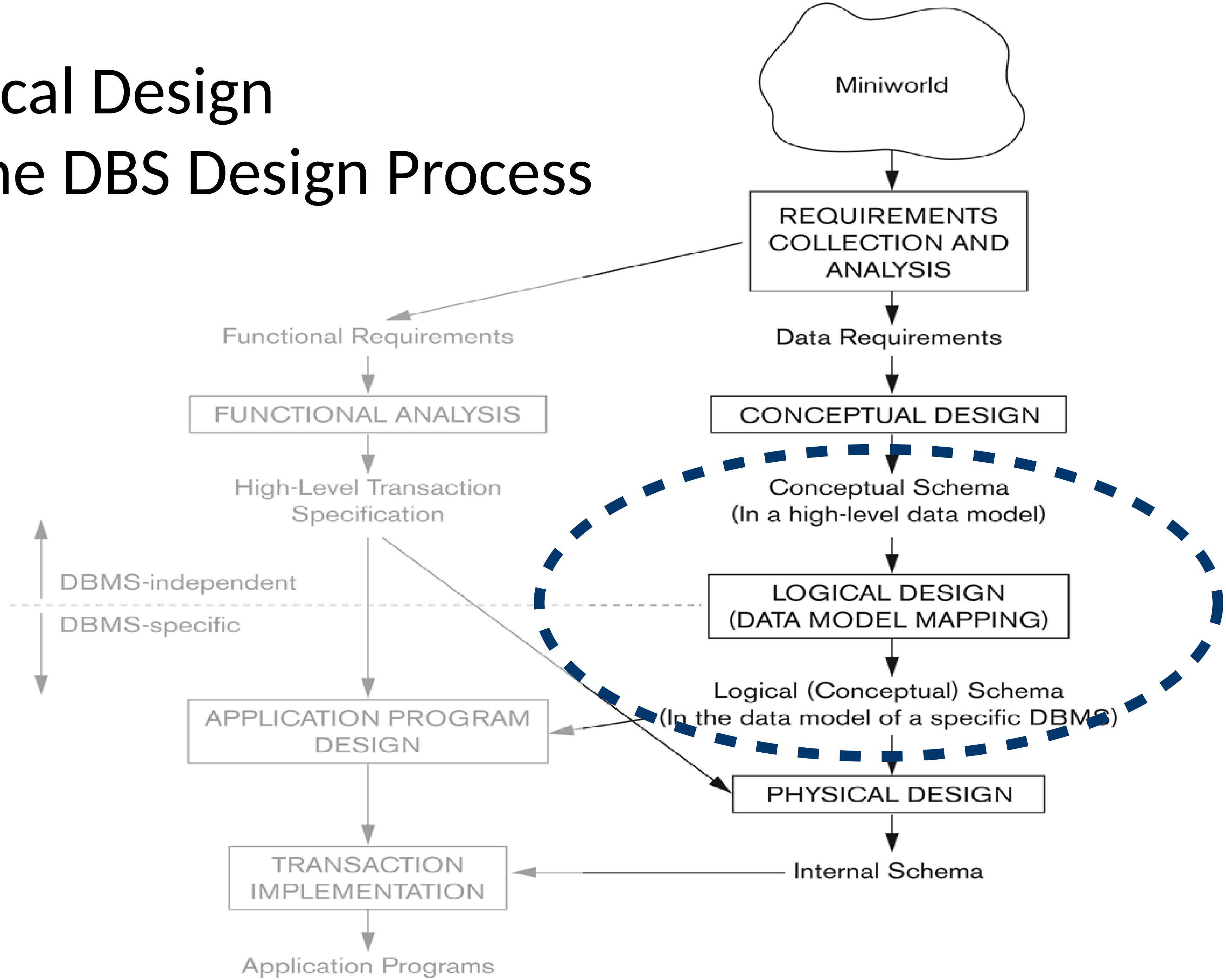
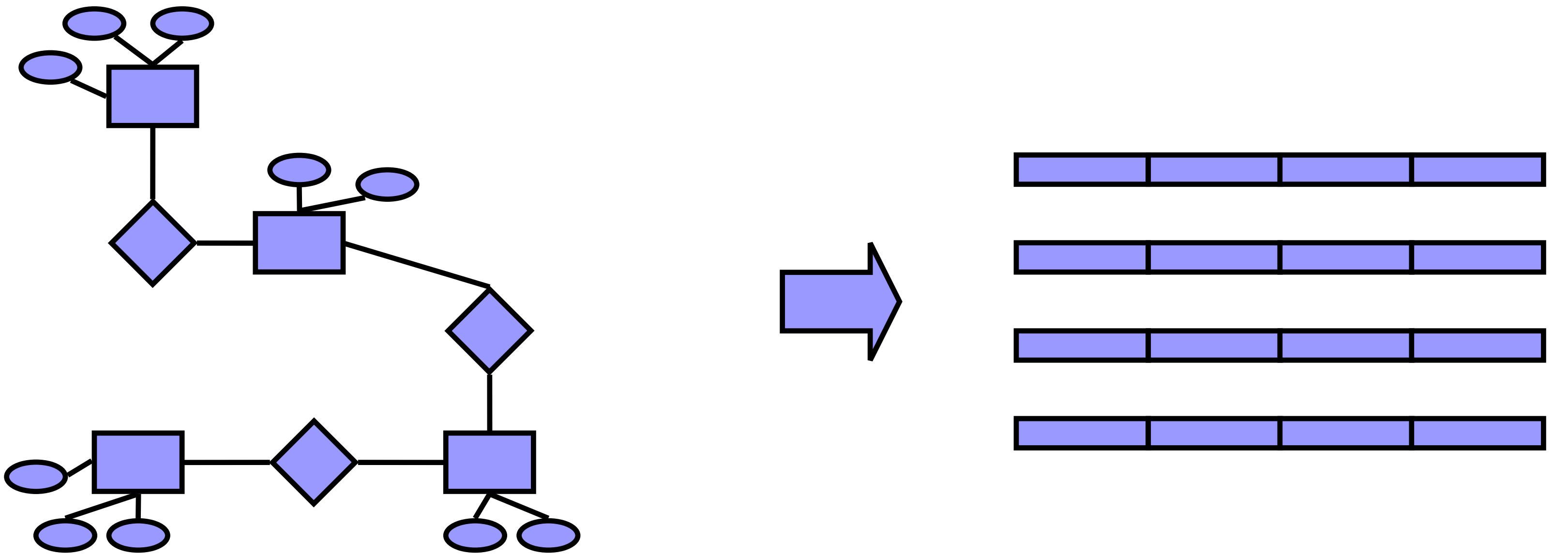


Figure from “Fundamentals of Database Systems” by Elmasri and Navathe, Addison Wesley.

Translation of ER to a Relational DB Schema

- Done by running an algorithm



SQL

Structured Query Language

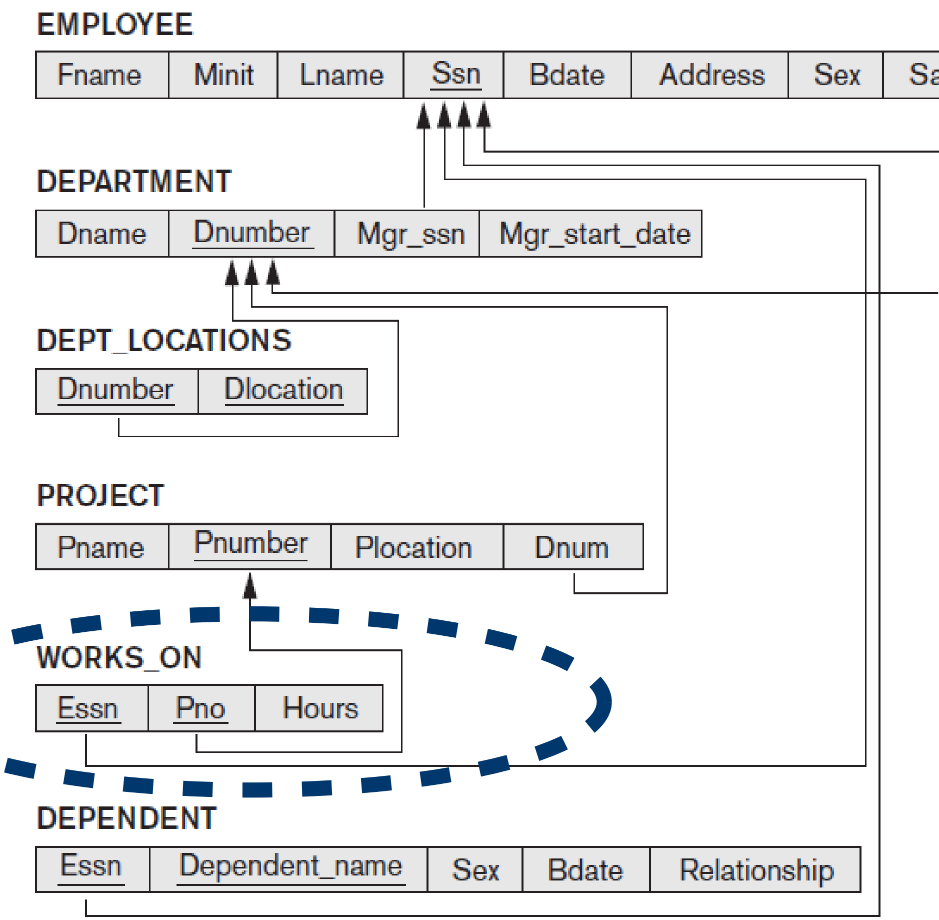
Main Characteristics and Features of SQL

- For defining and for manipulating relational databases
- **Declarative** language (what data to get, not how)
- Considered one of the major reasons for the commercial success of relational databases

Relational Model	SQL
relation	table
tuple	row
attribute	column

- Defining: CREATE TABLE, CREATE …, ALTER …, DROP ...
- Querying: SELECT
- Modifying: INSERT, UPDATE, DELETE
- Advanced features: triggers, stored procedures

Creating Tables (Example)




```
CREATE TABLE WORKS_ON (
   ESSN     integer,
   PNO      integer,
   HOURS    decimal(3,1),

   constraint pk_workson
     primary key (ESSN, PNO),

   constraint  fk_works_emp
     FOREIGN KEY (ESSN)
     references EMPLOYEE(SSN),

   constraint  fk_works_proj
     FOREIGN KEY (PNO)
     references PROJECT(PNUMBER) );
```

Example from “Fundamentals of Database Systems” by Elmasri and Navathe, Addison Wesley.

Example of an SQL Query

EMPLOYEE

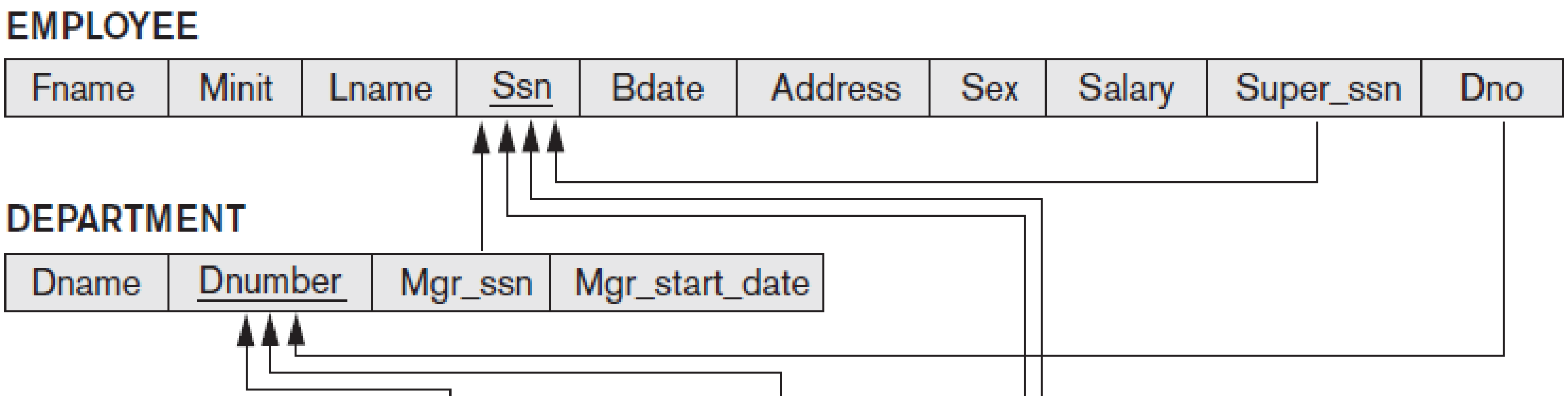
Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

- List the last name, birth date and address for all employees whose name is `Alicia J. Zelaya‘

SELECT *LNAME*, *BDATE*, *ADDRESS*
FROM *EMPLOYEE*
WHERE *FNAME* = ‘Alicia’
AND MINIT = ‘J’
AND LNAME = ‘Zelaya’**;**

Example from “Fundamentals of Database Systems” by Elmasri and Navathe, Addison Wesley.

Updating Data (Example)



- Give all employees in the ‘Research’ department a 10% raise in salary

UPDATE *EMPLOYEE*
SET *SALARY = SALARY*1.1*
WHERE *DNO IN* (**SELECT** *DNUMBER*
FROM *DEPARTMENT*
WHERE *DNAME = ‘Research’*);

Example from “Fundamentals of Database Systems” by Elmasri and Navathe, Addison Wesley.

Physical Design

Data Files and Indexes

Physical Design
in the DBS Design Process

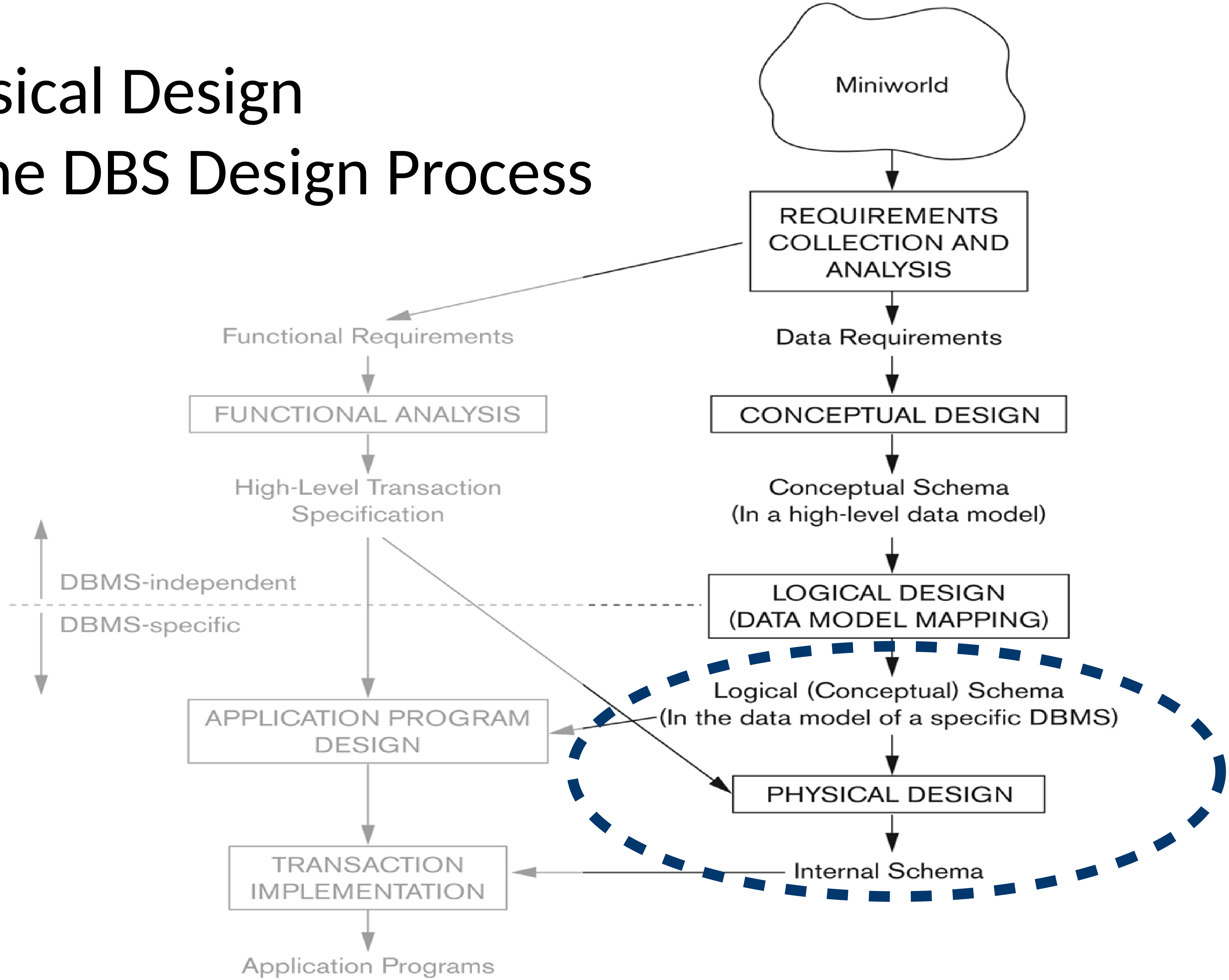
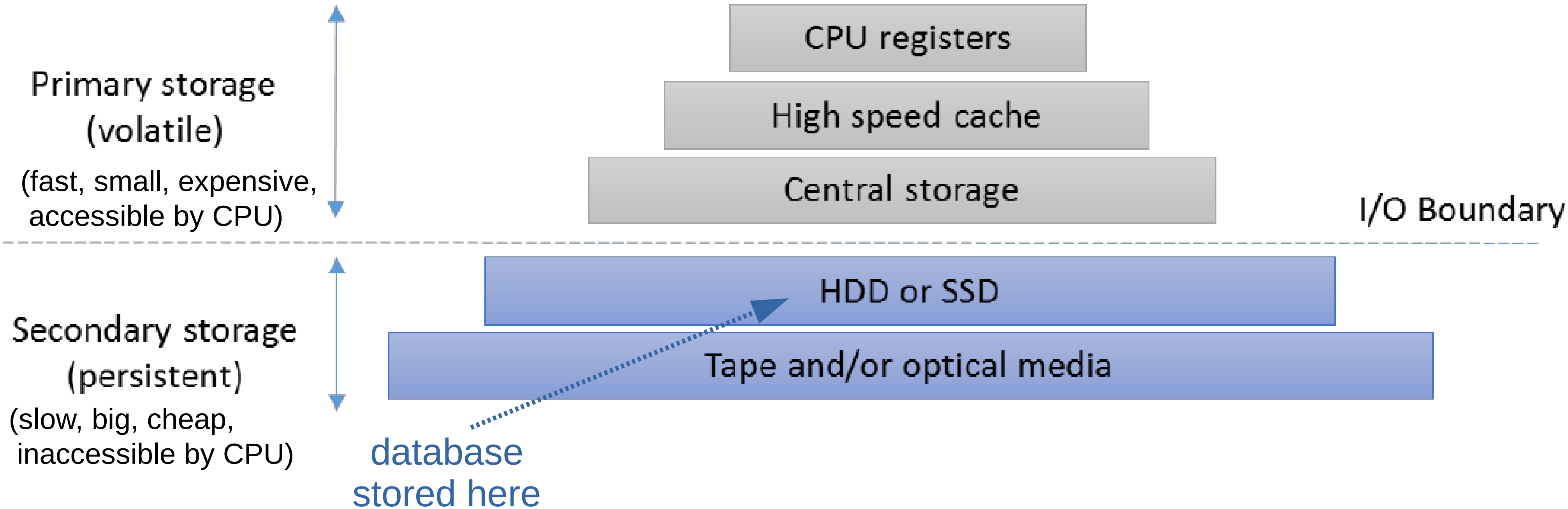


Figure from “Fundamentals of Database Systems” by Elmasri and Navathe, Addison Wesley.

Storage Hierarchy

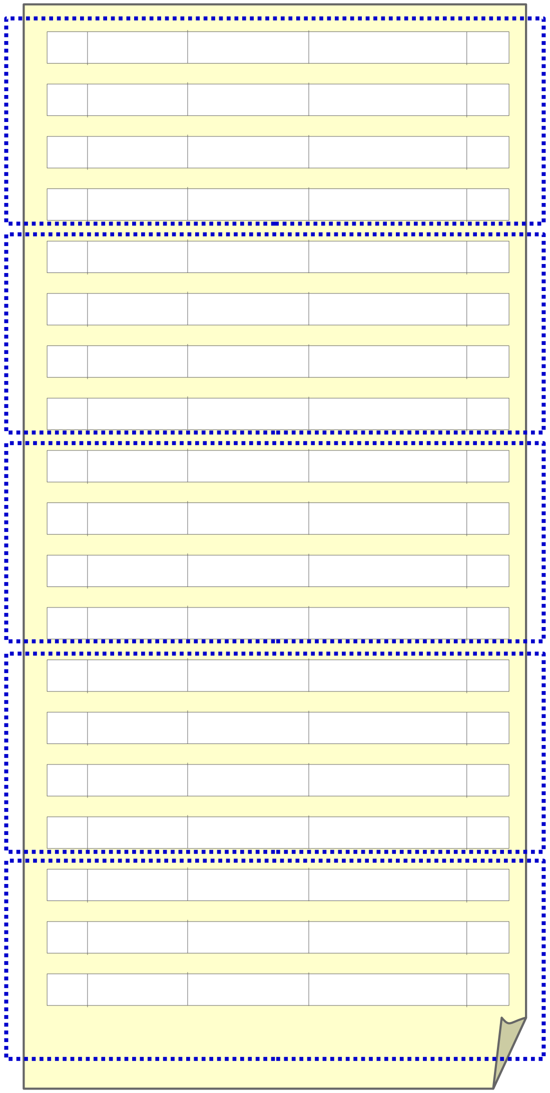


- Transferring data between secondary and primary storage is a major bottleneck!
  - CPU instruction: ca. 1 ns (10^{-9} secs)
  - Main memory access: ca. 10 ns (10^{-8} secs)
  - Disk access: ca. 1 ms (1M ns, 10^{-3} secs)

Image source: https://www.pdbmbook.com/vault/Chapter%20Physical%20File%20Organization%20and%20Indexing.pdf/access

Files and Records

- Data stored in files
  - file is a sequence of records
  - record is a set of field values
  - for instance, file = relation
    record = row
    field = attribute value
- HDDs formatted into equal-sized blocks
  - typical block sizes: 512 – 8192 bytes
  - such blocks are the unit of data transfer between disk and main memory
- Every file needs to be split into blocks
  - every record is allocated to one of these file blocks



Heap Files

- Records are added to the end of the file
- Adding a record is cheap
- Retrieving, removing, and updating a record is expensive because it implies *linear search*
  - average case: $\left\lceil \frac{b}{2} \right\rceil$ block accesses
  - worst case: b block accesses

(where b is the number of blocks of the file)

- Record removal also implies waste of space
  - periodic reorganization

Sorted Files

ID#	Name	Dept.	Salary
1	Daniels	12	2000
2	Lancaster	13	4000
3	Andersson		
4	Andersson		

Block 1

ID#	Name	Dept.	Salary
5	Silver		
6	Molin		
7	French		
8	Daniels		

Block 2

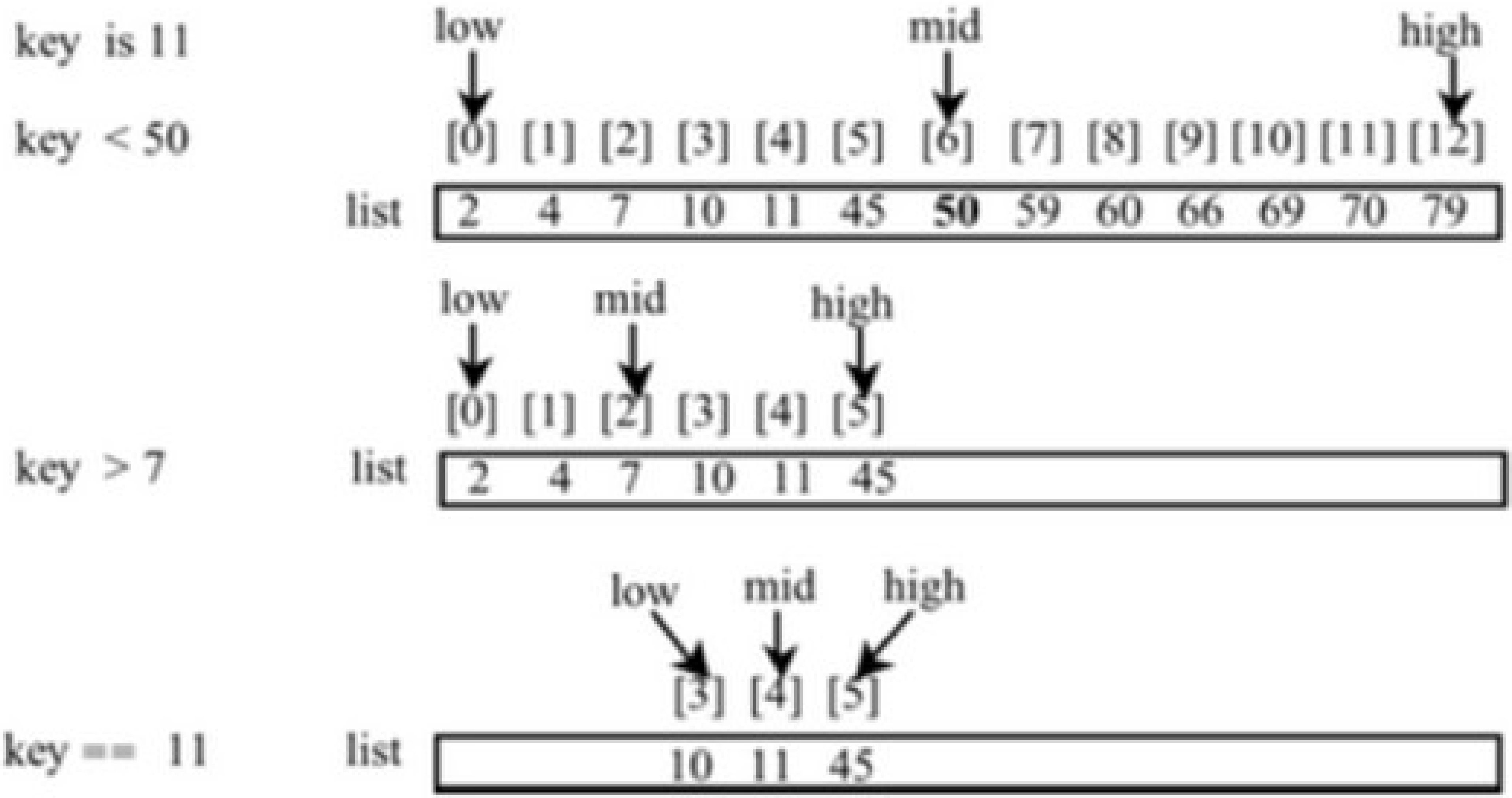
ID#	Name	Dept.	Salary
9	Andersson		
10	Hagberg		
11	Yang		
12	Miller		

Block 3

- Records ordered according to some field (e.g., ID# in the example)
- Ordered record retrieval is cheap (i.e., on the ordering field)
  - all records: access the blocks sequentially
  - next record: probably in the same block
  - random record: *binary search*; hence, $\lceil \log_2 b \rceil$ block accesses* in the worst case
- Retrieval based on any other field is still expensive
- Adding a record is expensive, but removing is less expensive (deletion markers, periodic reorganization)

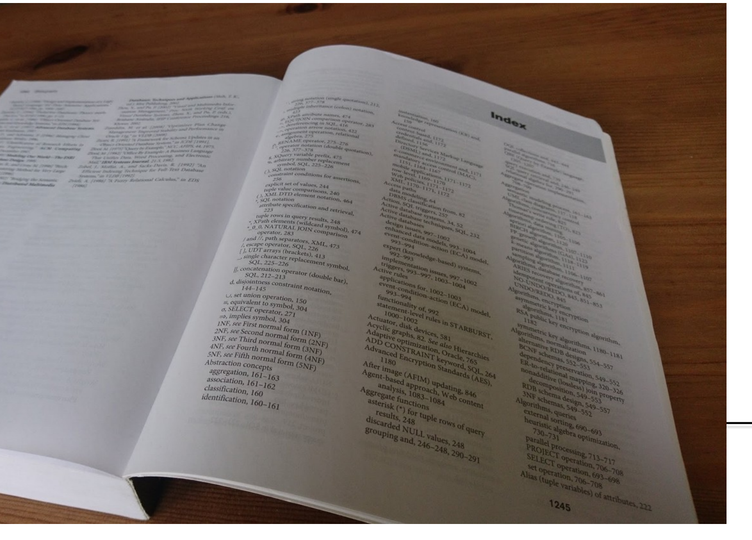
*recall, b is the number of blocks of the file

Binary Search



Indexes

- File organization (heap, sorted, hash) determines the primary method to access data in a file
  - e.g., sequential search, binary search
- However, this may not be fast enough
- Solution: index files
  - introduce secondary access methods
  - goal: speed up access under specific conditions
  - there exist various types of index structures

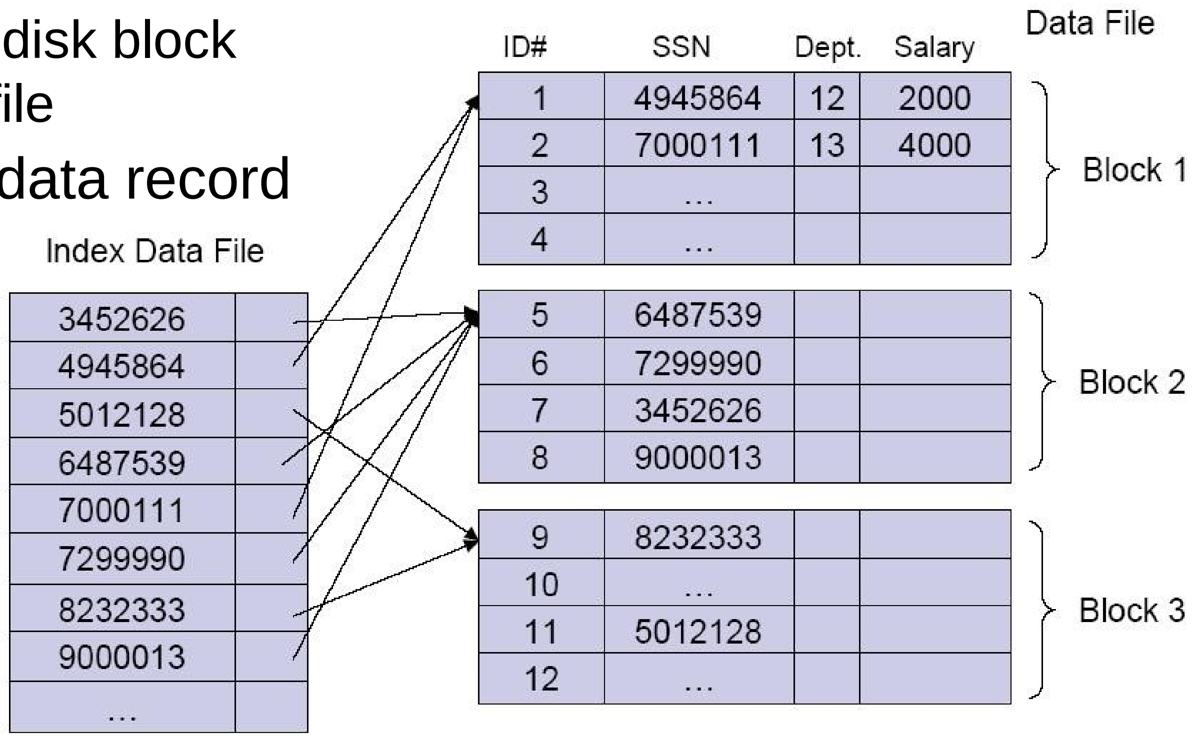


Example: Secondary Index on Key Field

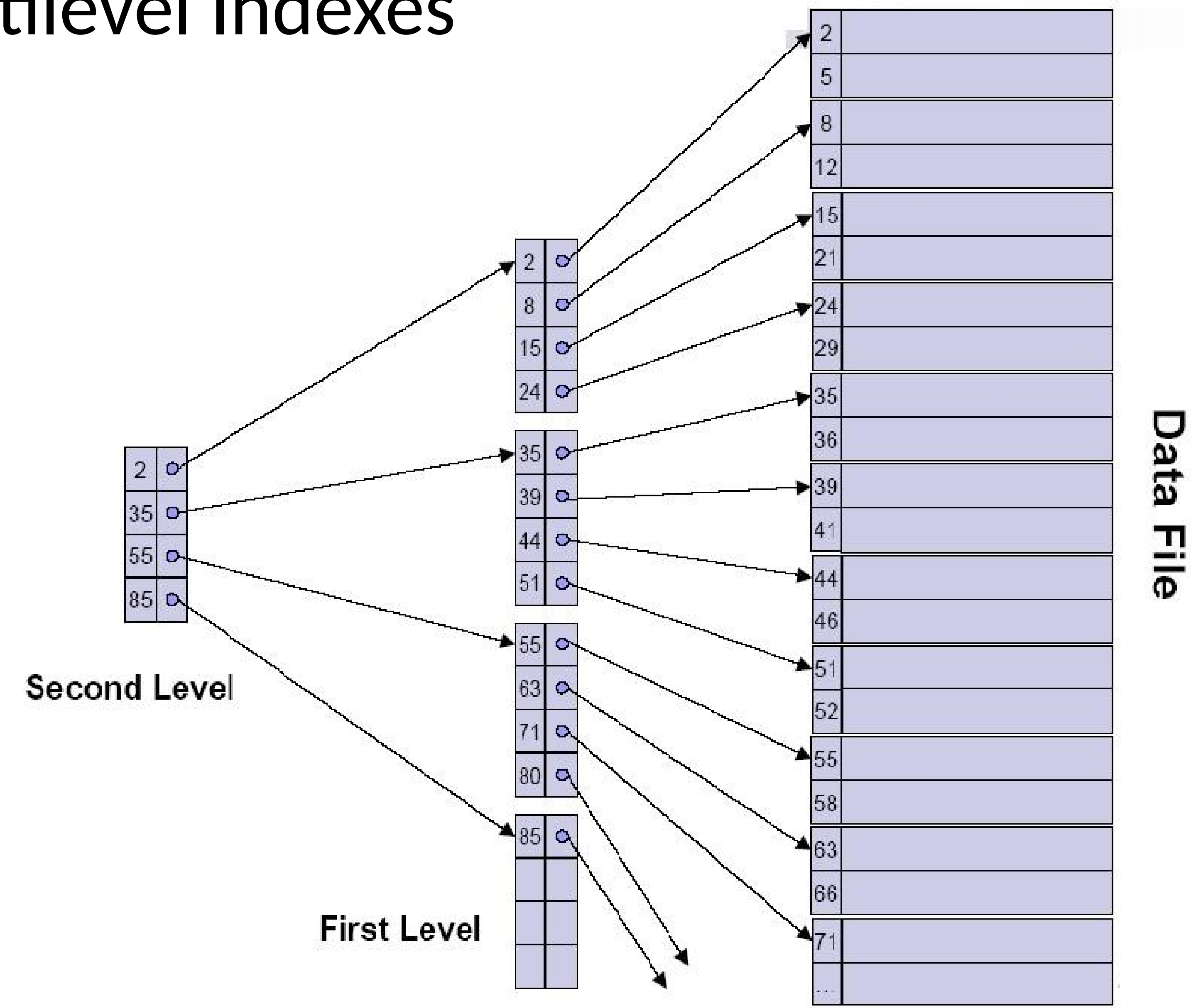
- Index on a *non*-ordering *key* field *F*
  - in the example, the SSN field is this field *F*
  - data file may be sorted or not (in the example, it is)
- Secondary index: additional *sorted* file
  - records in this file contain two fields:

    V - one of the values of *F*

    P - pointer to a disk block of the data file
  - one index record per data record
- Example index speeds up the search for records based on a given SSN value
- Index maintenance!



Multilevel Indexes

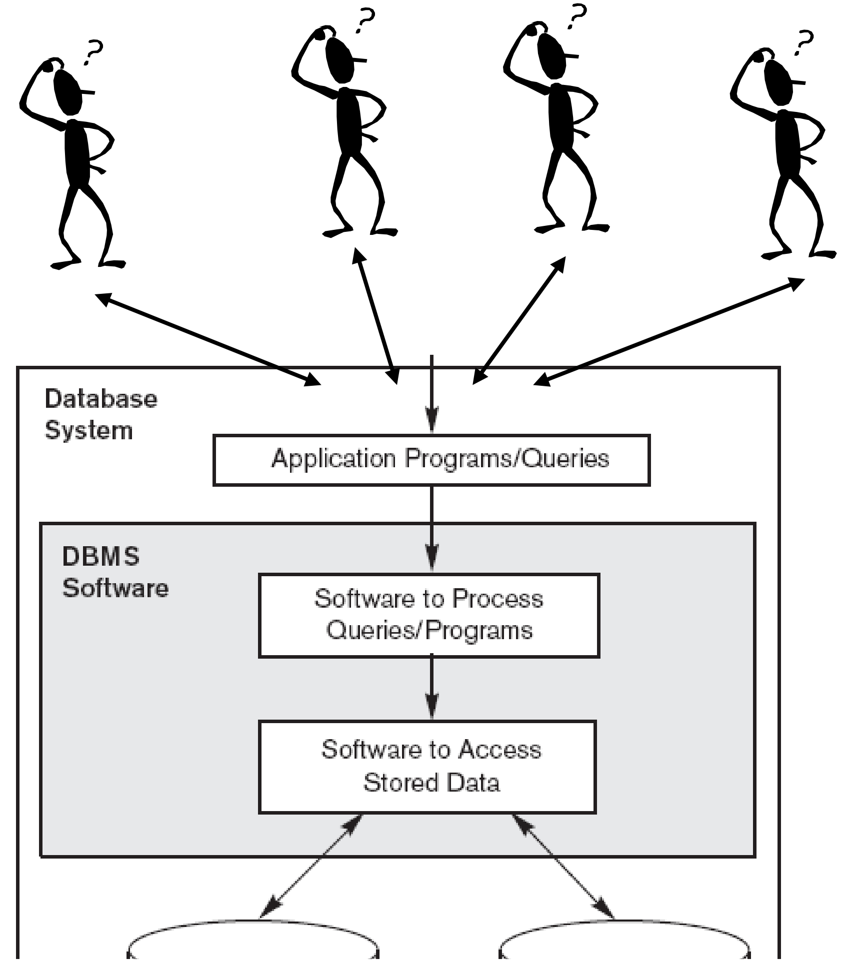


Transactions

for Concurrency Control and for Recovery

Motivation

- A DB is a shared resource accessed by many users and processes concurrently
- As any computer program, a DBMS may be disrupted by system or media failure
- Transaction processing is about avoiding problems caused by both
  - failure and
  - concurrency



1	read_item(savings);		a	read_item(checking);
2	savings = savings - $100;		b	checking = checking - $20;
3	write_item(savings);		c	write_item(checking);
4	read_item(checking);		d	dispense $20 to customer;
5	checking = checking + $100;			
6	write_item(checking);			

Transaction

- An application-specified, *atomic* and *durable* unit of work (a process) that comprises one or more database access operations
- Characteristic operations
  - Read (database retrieval, such as SQL SELECT)
  - Write (modify DB, such as INSERT, UPDATE, DELETE)

	Transaction 1			**Transaction 2**
1	read_item(savings);		a	read_item(checking);
2	savings = savings - $100;		b	checking = checking - $20;
3	write_item(savings);		c	write_item(checking);
4	read_item(checking);		d	dispense $20 to customer;
5	checking = checking + $100;			
6	write_item(checking);			

Desirable Properties of Transactions (ACID)

- **A**tomicity: a transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all
- **C**onsistency preservation: a correct execution of a TA must take the DB from one consistent state to another
- **I**solation: even though TAs are executing concurrently, they should appear to be executed in isolation; that is, their final effect should be as if each TA was executed alone from start to end
- **D**urability: once a TA is committed, its changes applied to the database must never be lost due to subsequent failure

- Who ensures that the ACID properties are satisfied?
  - Consistency pres.: Constraint subsystem (and programmer)
  - Atomicity and Durability: Recovery subsystem
  - Isolation: Concurrency control subsystem

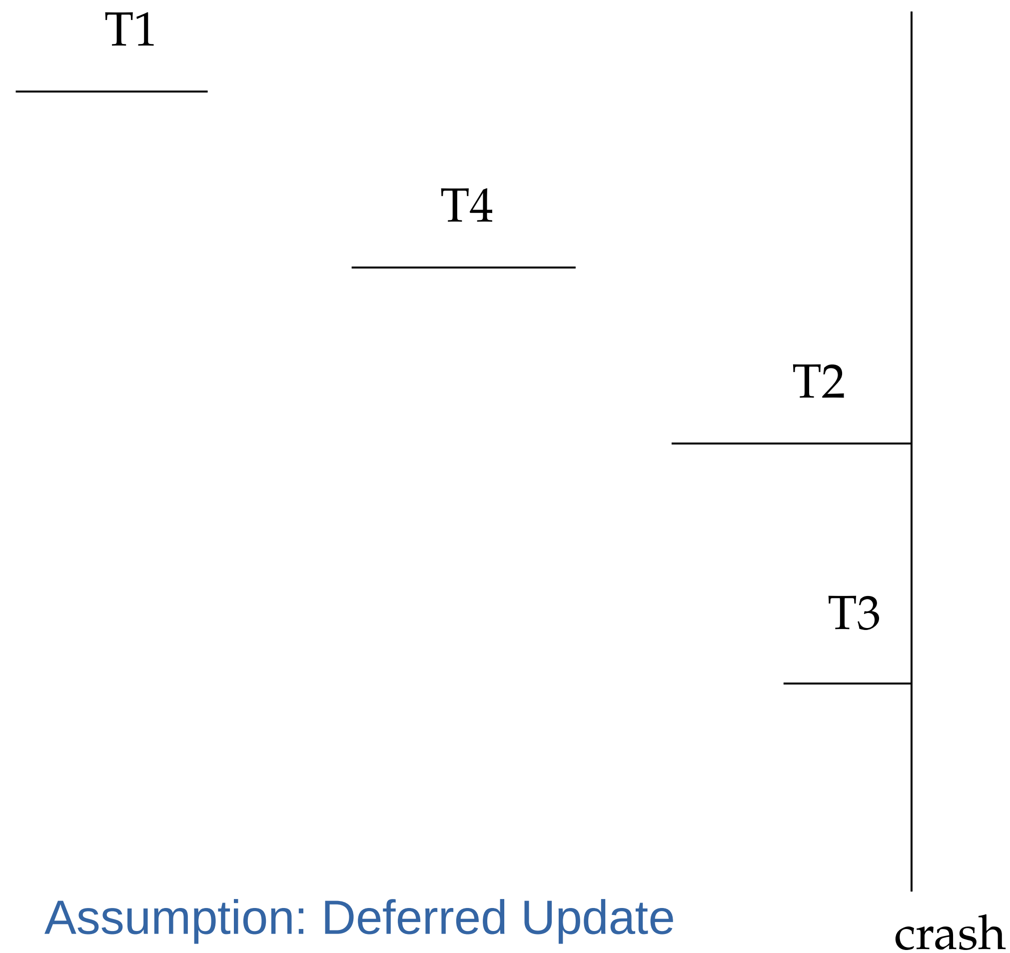
Concurrency Control (to Ensure Isolation)

A transaction follows the **two-phase locking (2PL) protocol** if *all* of its read_lock() and write_lock() operations come before its *first* unlock() operation.

T1 ✓ follows 2PL	T2 violates 2PL
Read-lock(my-account1)	Read-lock(my-account1)
Read-item(my-account1)	Read-item(my-account1)
Write-lock(my-account2)	Unlock(my-account1)
Unlock(my-account1)	Write-lock(my-account2)
Read-item(my-account2)	Read-item(my-account2)
my-account2 := my-account2 + 2000	my-account2 := my-account2 + 2000
Write-item(my-account2)	Write-item(my-account2)
Unlock(my-account2)	Unlock(my-account2)

Log-Based Recovery (to Ensure Atomicity and Durability)

start-transaction T1
write-item T1, D, 10, 20
commit T1
start-transaction T4
write-item T4, B, 10, 20
write-item T4, A, 5, 10
commit T4
start-transaction T2
write-item T2, B, 20, 15 *(ignore)*
start-transaction T3
write-item T3, A, 10, 30 *(ignore)*
write-item T2, D, 20, 25 *(ignore)*
CRASH



Assumption: Deferred Update

NO-UNDO

REDO: T1, T4

Summary and Outlook

Summary and Outlook

- Focus on relational databases
  - Entity-relationship model (ER diagrams)
  - Relational model (relation schemas, constraints, SQL)
  - Physical model (files and indexes)
  - Transactions for concurrency control and for recovery
- Many more details in course “Database Technology” (TDDD12, TDDD37, TDDD81, 732A57)
- Other types of databases and data models
  - NoSQL, graph databases, XML, ontologies
  - Course “Advanced Data Models and Databases” (TDDD43)
- Thesis projects on DB topics available

www.liu.se