Table of Contents

Chapter 1: Introduction.....2

Chapter 2: File System.....3

Chapter 3: Data Models.....6

Chapter 4: Relational Databases.....9

Chapter 5: Entity Relationship Model.....16

Chapter 6: Normalization.....24

Chapter 7: SQL.....30

Chapter 8: Extended Entity Relationship Model.....36

Chapter 9: Database Dev Life Cycles.....41

Chapter 1: Introduction

Data: Raw facts, building blocks of information. Unprocessed Information.

Information: Data processed to reveal meaning.

Database—shared, integrated computer structure that stores:

- End user data (raw facts)
- Metadata (data about data)

DBMS (database management system):

- Collection of programs that manages database structure and controls access to data
- Possible to share data among multiple applications or users
- Makes data management more efficient and effective

Advantages of DBMS:

- End users have better access to more and better managed data.
- Promotes integrated view of organization's operations.
- The probability of data inconsistency is greatly reduced.
- Possible to produce quick answers to ad hoc queries.

Types of Databases:

Single user: Supports only one user at a time

Desktop: Single-user database running on a personal computer

Multi-user: Supports multiple users at the same time

Workgroup: Multi-user database that supports a small group of users or a single department

Enterprise: Multi-user database that supports a large group of users or an entire organization

Can be classified by location:

Centralized: Supports data located at a single site

Distributed: Supports data distributed across several sites

Can be classified by use:

Transactional (or production): Supports a company's day-to-day operations

Data warehouse: Stores data used to generate information required to make tactical or strategic decisions

Often used to store historical data

Structure is quite different

QUESTIONS FOR TOPIC 1:

What is the difference between data and information: Data is raw, unprocessed information. Whereas information is processed information that has meaning to it.

Describe the difference between database and DBMS: Database is shared, integrated computer structure that stores end user data and metadata. While DBMS is a collection of programs that manages database structure and controls access to data

Briefly explain 3 types of databases:

- **Single use**r: Supports only one user at a time
- **Desktop:** Single-user database running on a personal computer
- **Multi-user:** Supports multiple users at the same time

Chapter 2: File System

Why is database design important?

It defines the database's expected use; it also highlights different approaches needed for different types of databases. It helps avoid redundant data, which can lead to bad decision and organization failures.

File Systems:

Managing data with file systems is obsolete!

Manual File systems:

- Collection of file folders kept in file cabinet.
- Organization within folders based on data's expected use (ideally logically related)
- System adequate for small amounts of data with few reporting requirements **(DISADVANTAGE)**
- Finding and using data in growing collections of file folders became time-consuming and cumbersome. **(DISADVANTAGE)**

Conversion from manual to computer system:

- Could be technically complex, requiring hiring of data processing (DP) specialists.
- Resulted in numerous “home-grown” systems being created.

Initially, computer files were similar in design to manual files.

Field: A character or group of characters (alphabetical or numerical) that has a specific meaning. A field is used to define and store data.

Record: A logically connected set of one or more fields that describes a person, place, or thing.

File: A collection of related Records.

Disadvantages of File Systems:

- Time-consuming, high-level activity.
- As the number of files expands, system administration becomes difficult.
- Making changes in existing file structure is difficult.
- File structure changes require modifications in all programs that use data in that file.
- Modifications are likely to produce errors, requiring additional time to “debug” the program.
- Security features are hard to program and therefore often omitted.

Data redundancy results in data inconsistency, where different and conflicting versions of the same data appear in different places. Errors more likely to occur when complex entries are made in several different files and/or recur frequently in one or more files.

Types of data anomalies:

Update anomalies: Occur when changes must be made to existing records.

Insertion anomalies: Occur when entering new records.

Deletion anomalies: Occur when deleting records.

Problems inherent in file systems make using a database system desirable:

- **File system**
  - Many separate and unrelated files.
- **Database**
  - Logically related data stored in a single logical data repository.

Database system is composed of five main parts:

- Hardware
- Software
- Operating system software
- DBMS software
- Application programs and utility software
- People
- Procedures
- Data

DBMS performs functions that guarantee integrity and consistency of data:

- **Data dictionary management:** Defines data elements and their relationships.
- **Data storage management:** Stores data and related data entry forms, report definitions, etc.
- **Data transformation and presentation:** Translates logical requests into commands to physically locate and retrieve the requested data.
- **Security management:** Enforces user security and data privacy within database.
- **Multiuser access control:** Uses sophisticated algorithms to ensure multiple users can access the database concurrently without compromising the integrity of the database.
- **Backup and recovery management:** Provides backup and data recovery procedures.
- **Data integrity management:** Promotes and enforces integrity rules.
- **Database access languages and application programming interfaces:** Provide data access through a query language.
- **Database communication interfaces:** Allow database to accept end-user requests via multiple, different network environments.

Questions for Topic 2:

What are the disadvantages of file-based system:

a) Time-consuming, high-level activity.

b) As the number of files expands, system administration becomes difficult.

c) Making changes in existing file structure is difficult.

What is data redundancy?

Where different and conflicting versions of the same data appear in different places. Errors more likely to occur when complex entries are made in several different files and/or recur frequently in one or more files.

Explain different types of data anomaly:

Update anomalies: Occur when changes must be made to existing records.

Insertion anomalies: Occur when entering new records.

Deletion anomalies: Occur when deleting records.

Describe 3 functions of DBMS:

Data dictionary management: Defines data elements and their relationships.

Data storage management: Stores data and related data entry forms, report definitions, etc.

Security management: Enforces user security and data privacy within database.

Chapter 3: Data Models

Data models:

Relatively simple representations, usually graphical, of complex real-world data structures.

Facilitate interaction among the designer, the applications programmer, and the end user.

End-users have different views and needs for data.

Data model organizes data for various users.

Data model building blocks:

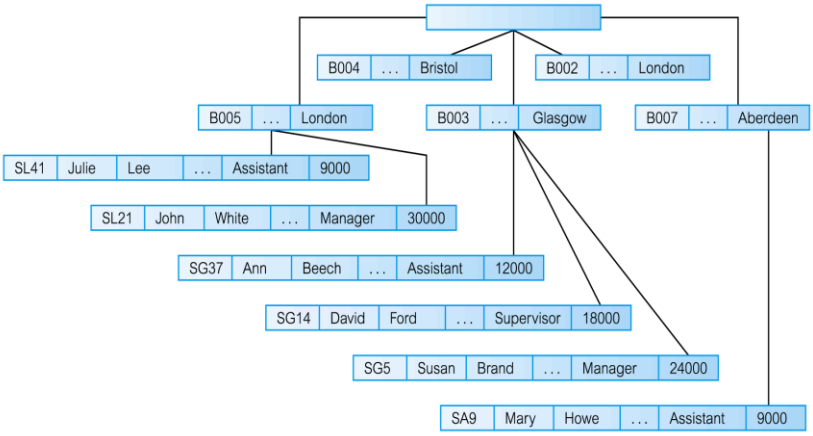
- **Entity -** anything about which data are to be collected and stored.
- **Attribute -** a characteristic of an entity
- **Relationship -** describes an association among entities.
  - One-to-many (1:M) relationship
  - Many-to-many (M:N or M:M) relationship
  - One-to-one (1:1) relationship
  - Relationships are bi-directional
- **Constraint -** a restriction placed on the data.

- **Business rules:** Brief, precise, and unambiguous. Apply to any organization that stores and uses data to generate information. Must be rendered in writing and must be kept up to date. Describe characteristics of the data as viewed by the company. Sometimes are external to the organization.

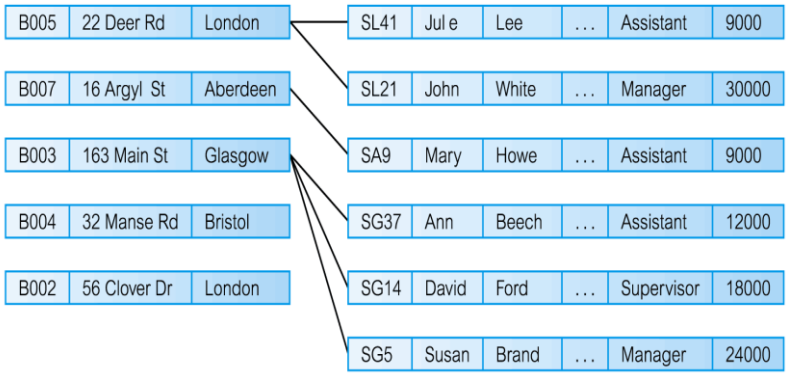
Sources of Business Rules:

- Company managers
- Policy makers
- Department managers
- Written documentation
  - Procedures
  - Standards
  - Operations manuals
- Direct interviews with end users

Hierarchical Data Model:



Network Model:



Relational Model:

Relational Table: Stores a collection of related entities where it resembles a file.

Relational table is purely logical structure: How data are physically stored in the database is of no concern to the user or the designer. This property became the source of a real database revolution.

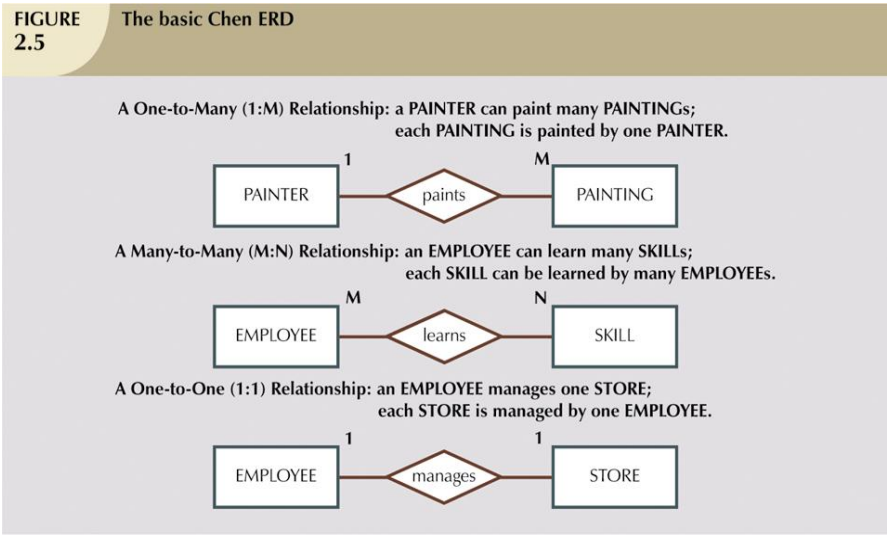
Entity Relationship Diagram (ERD)

Introduced by Chen in 1976

- **Entity relationship diagram (ERD)**
  - Uses graphic representations to model database components
  - Entity is mapped to a relational table
- **Entity instance (or occurrence) is row in table.**
- **Entity set is collection of like entities.**
- **Connectivity label types of relationships**
  - Diamond connected to related entities through a relationship line

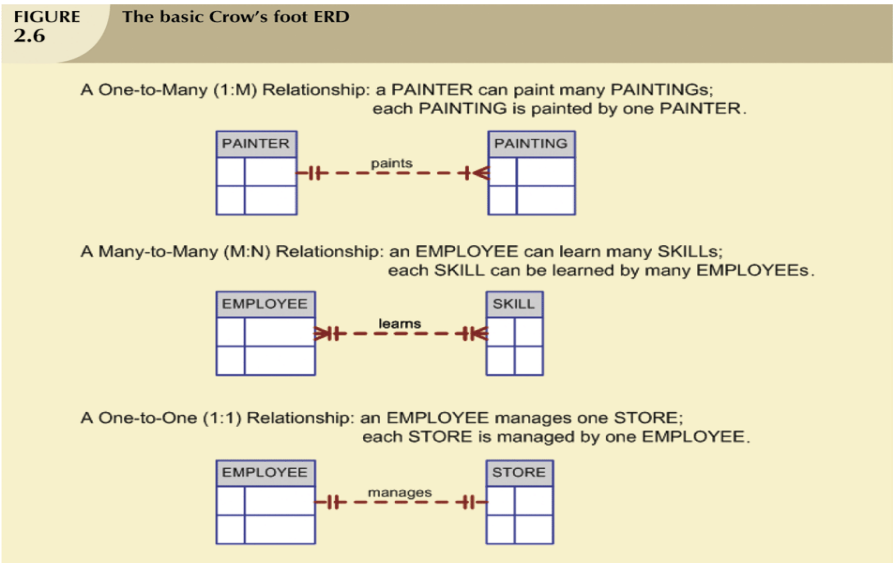
CHEN ERD

FIGURE 2.5 The basic Chen ERD



CROW'S ERD

FIGURE 2.6 The basic Crow's foot ERD



Questions for Topic 3:

What is data model?

Relatively simple representations, usually graphical, of complex real-world data structures.

Facilitate interaction among the designer, the applications programmer, and the end user.

What is the source of business rules?

- Company managers
- Policy makers
- Department managers
- Written documentation
- Direct interviews with end users

What is entity relationship model?

Entity relationship diagram (ERD) is a data model which uses graphic representations to model database components. Entity is mapped to a relational table.

Chapter 4: Relational Databases

- **Relational model**
  - Enables programmers to view data logically rather than physically.
- **Table**
  - Has advantages of structural and data independence?
  - Resembles a file from conceptual point of view.
  - Easier to understand than its hierarchical and network database predecessors.
  - Each table row is called a **Tuple.**
  - Each column represents and attributes.
  - Each row and column represent a single data value.
  - Each column has a specific range of values known as the **attribute domain.**

Table: two-dimensional structure composed of rows and columns

- Contains group of related entities = an entity set
- Terms entity set and table are often used interchangeably.

Table also called a relation because the **relational model's creator, Codd**, used the term relation as a synonym for table

Think of a table as a persistent relation: A relation whose contents can be permanently saved for future use.

Keys: Consists of one or more attributes that determine other attributes.

- **Primary key (PK) is an attribute** (or a combination of attributes) that uniquely identifies any given entity (row)
- **Key's role is based on determination.**
  - If you know the value of attribute A, you can look up (determine) the value of attribute B.
- **Foreign key (FK)**
  - An attribute whose values match primary key values in the related table.
- **Referential integrity**
  - FK contains a value that refers to an existing valid tuple (row) in another relation.
- **Secondary key**
  - Key used strictly for data retrieval purposes.
- **Candidate key**
  - A Candidate Key can be any column or a combination of columns that can qualify as unique key in database.
  - There can be multiple Candidate Keys in one table.
  - Each Candidate Key can qualify as Primary Key.

How to select a Primary Key:

- Select a key that does not contain NULL
- Select a key that is unique and does not repeat
- Make sure that Primary Key does not keep changing

Nulls: No data entry**,** not permitted in primary key**,** should be avoided in other attributes.

- Can represent
  - An unknown attribute value
  - A known, but missing, attribute value
  - A “not applicable” condition
- Can create problems when functions such as COUNT, AVERAGE, and SUM are used
- Can create logical problems when relational tables are linked

Controlled redundancy:

- Makes the relational database work.
- Tables within the database share common attributes that enable the tables to be linked together.

FIGURE 3.2 An example of a simple relational database

Table name: PRODUCT
Primary key: PROD_CODE
Foreign key: VEND_CODE

Database name: Ch03_SaleCo

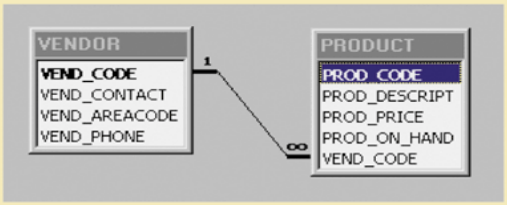
PROD_CODE	PROD_DESCRIPT	PROD_PRICE	PROD_ON_HAND	VEND_CODE
001278-AB	Claw hammer	$12.95	23	232
123-21UUY	Houselite chain saw, 16-in. bar	$189.99	4	235
QER-34256	Sledge hammer, 16-lb. head	$18.63	6	231
SRE-657UG	Rat-tail file	$2.99	15	232
ZZX/3245Q	Steel tape, 12-ft. length	$6.79	8	235

link

Table name: VENDOR
Primary key: VEND_CODE
Foreign key: none

VEND_CODE	VEND_CONTACT	VEND_AREACODE	VEND_PHONE
230	Shelly K. Smithson	608	555-1234
231	James Johnson	615	123-4536
232	Annelise Crystall	608	224-2134
233	Candice Wallace	904	342-6567
234	Arthur Jones	615	123-3324
235	Henry Ortozo	615	899-3425

FIGURE 3.3 The relational diagram for the Ch03_SaleCo database



Data dictionary

- Provides detailed accounting of all tables found within the user/designer-created database
- Contains (at least) all the attribute names and characteristics for each table in the system
- Contains metadata—data about data

- Sometimes described as “the database designer’s database” because it records the design decisions about tables and their structures

TABLE 3.6 A Sample Data Dictionary

TABLE NAME	ATTRIBUTE NAME	CONTENTS	TYPE	FORMAT	RANGE	REQUIRED	PK OR FK	FK REFERENCED TABLE
CUSTOMER	CUS_CODE	Customer account code	CHAR(5)	99999	10000–99999	Y	PK	
	CUS_LNAME	Customer last name	VCCHAR(20)	Xxxxxxxx		Y		
	CUS_FNAME	Customer first name	VCHAR(20	Xxxxxxxx		Y		
	CUS_INITIAL	Customer initial	CHAR(1)	X	100–999		FK	AGENT
	CUS_RENEW_DATE	Customer insurance renewal date	DATE	dd-mmm-yyyy				
	AGENT_CODE	Agent code	CHAR(3)	999				
AGENT	AGENT_CODE	Agent code	CHAR(3)	999		Y	PK	
	AGENT_AREACODE	Agent area code	CHAR(3)	999		Y		
	AGENT_PHONE	Agent telephone number	CHAR(8)	999-9999	0.00–9,999,999.99	Y		
	AGENT_LNAME	Agent last name	VCHAR(20)	Xxxxxxxx		Y		
	AGENT_YTD_SLS	Agent year-to-date sales	NUMBER(9,2)	9,999,999.99		Y		

FK = Foreign key
PK = Primary key
CHAR = Fixed character length data (1–255 characters)
VARCHAR = Variable character length data (1–2,000 characters)
NUMBER = Numeric data (NUMBER(9,2) is used to specify numbers with two decimal places and up to nine digits, including the decimal places. Some RDBMSs permit the use of a MONEY or CURRENCY data type.)

Relationships within the Relational Database

1:M relationship

- Relational modeling ideal
- Should be the norm in any relational database design.

1:1 relationship

- Should be rare in any relational database design.

M:N relationships

- Cannot be implemented as such in the relational model.
- M:N relationships can be changed into two 1:M relationships

1:M relationship:

- Relational database norm
- Found in any database environment.

FIGURE 3.18 The 1:M relationship between PAINTER and PAINTING

FIGURE 3.19 The implemented 1:M relationship between PAINTER and PAINTING

Table name: PAINTER
Primary key: PAINTER_NUM
Foreign key: none
Database name: Ch03_Museum

PAINTER_NUM	PAINTER_LNAME	PAINTER_FNAME	PAINTER_INITIAL
123	Ross	Georgette	P
126	Itero	Julio	G

Table name: PAINTING
Primary key: PAINTING_NUM
Foreign key: PAINTER_NUM

PAINTING_NUM	PAINTING_TITLE	PAINTER_NUM
1338	Dawn Thunder	123
1339	Vanilla Roses To Nowhere	123
1340	Tired Flounders	126
1341	Hasty Exit	123
1342	Plastic Paradise	126

Data redundancy leads to data anomalies.

- Such anomalies can destroy the effectiveness of the database.

Foreign keys

- Control data redundancies by using common attributes shared by tables.
- Crucial to exercising data redundancy control.

Sometimes, data redundancy is necessary.

1:1 relationship:

- One entity can be related to only one other entity, and vice versa.
- Sometimes means that entity components were not defined properly.
- Could indicate that two entities belong to the same table.
- As rare as a 1:1 relationship should be, certain conditions absolutely require their use.

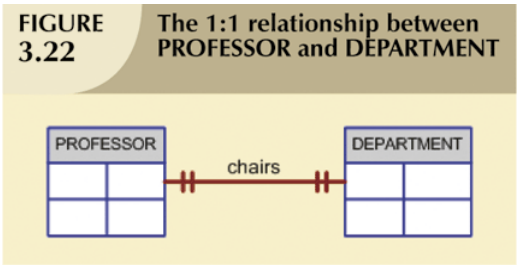


FIGURE 3.23 The implemented 1:1 relationship between PROFESSOR and DEPARTMENT

Table name: PROFESSOR
Primary key: EMP_NUM
Foreign key: DEPT_CODE

Database name: Ch03_TinyCollege

EMP_NUM	DEPT_CODE	PROF_OFFICE	PROF_EXTENSION	PROF_HIGH_DEGREE
103	HIST	DRE 156	6783	Ph.D.
104	ENG	DRE 102	5561	MA
105	ACCT	KLR 229D	8665	Ph.D.
106	MKTMGT	KLR 126	3899	Ph.D.
110	BIOL	AAK 160	3412	Ph.D.
114	ACCT	KLR 211	4436	Ph.D.
155	MATH	AAK 201	4440	Ph.D.
160	ENG	DRE 102	2248	Ph.D.
162	CIS	KLR 203E	2359	Ph.D.
191	MKTMGT	KLR 409B	4016	DBA
195	PSYCH	AAK 297	3550	Ph.D.
209	CIS	KLR 333	3421	Ph.D.
228	CIS	KLR 300	3000	Ph.D.
297	MATH	AAK 194	1145	Ph.D.
299	ECONFIN	KLR 284	2851	Ph.D.
301	ACCT	KLR 244	4683	Ph.D.
335	ENG	DRE 208	2000	Ph.D.
342	SOC	BBG 208	5514	Ph.D.
367	BIOL	AAK 230	8665	Ph.D.
401	HIST	DRE 156	6783	MA
425	ECONFIN	KLR 284	2851	MBA
435	ART	BBG 185	2278	Ph.D.

The 1:M DEPARTMENT employs PROFESSOR relationship is implemented through the placement of the DEPT_CODE foreign key in the PROFESSOR table.

The 1:1 PROFESSOR chairs DEPARTMENT relationship is implemented through the placement of the EMP_NUM foreign key in the DEPARTMENT table.

Table name: DEPARTMENT
Primary key: DEPT_CODE
Foreign key: EMP_NUM

DEPT_CODE	DEPT_NAME	SCHOOL_CODE	EMP_NUM	DEPT_ADDRESS	DEPT_EXTENSION
ACCT	Accounting	BUS	114	KLR 211, Box 52	3119
ART	Fine Arts	A&SCI	435	BBG 185, Box 128	2278
BIOL	Biology	A&SCI	387	AAK 230, Box 415	4117
CIS	Computer Info. Systems	BUS	209	KLR 333, Box 56	3245
ECONFIN	Economics/Finance	BUS	299	KLR 284, Box 63	3126
ENG	English	A&SCI	160	DRE 102, Box 223	1004
HIST	History	A&SCI	103	DRE 156, Box 284	1867
MATH	Mathematics	A&SCI	297	AAK 194, Box 422	4234
MKTMGT	Marketing/Management	BUS	106	KLR 126, Box 55	3342
PSYCH	Psychology	A&SCI	195	AAK 297, Box 438	4110
SOC	Sociology	A&SCI	342	BBG 208, Box 132	2008

1:M relationship:

- Relational database norm.
- Found in any database environment.

FIGURE 3.24 The ERD's M:N relationship between STUDENT and CLASS

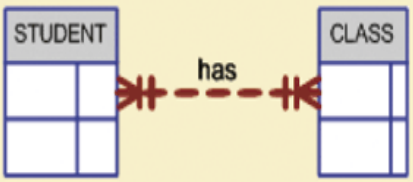


FIGURE 3.25 The M:N relationship between STUDENT and CLASS

Table name: STUDENT
Primary key: STU_NUM
Foreign key: none
Database name: Ch03_CollegeTry

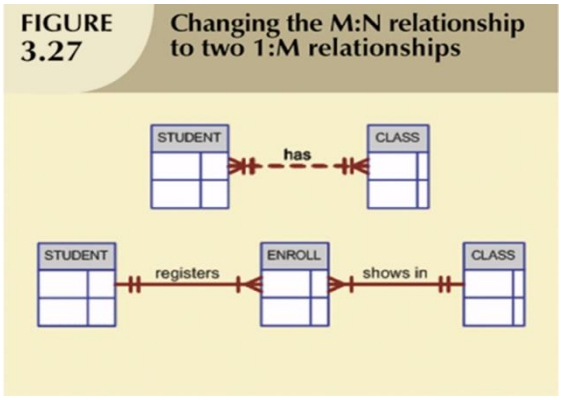
STU_NUM	STU_LNAME	CLASS_CODE
321452	Bowser	10014
321452	Bowser	10018
321452	Bowser	10021
324257	Smithson	10014
324257	Smithson	10018
324257	Smithson	10021

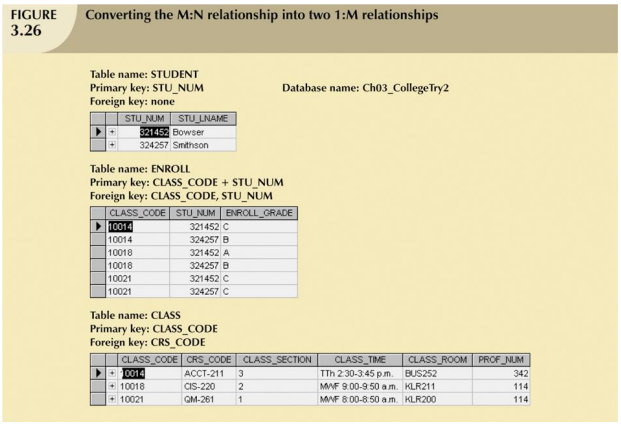
Table name: CLASS
Primary key: CLASS_CODE
Foreign key: STU_NUM

CLASS_CODE	STU_NUM	CRS_CODE	CLASS_SECTION	CLASS_TIME	CLASS_ROOM	PROF_NUM
10014	321452	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10014	324257	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10018	321452	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10018	324257	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10021	321452	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114
10021	324257	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114

Converting M:M to 1:M

- Can be implemented by breaking it up to produce a set of 1:M relationships
- Can avoid problems inherent to M:N relationship by creating a composite entity or bridge entity
- Implementation of a composite entity
- Yields required M:N to 1:M conversion
- Composite entity table must contain at least the primary keys of original tables
- Linking table contains multiple occurrences of the foreign key values
- Additional attributes may be assigned as needed





Chapter 4 Questions:

- **What is a candidate key?**

  A Candidate Key can be any column or a combination of columns that can qualify as unique key in database.

- **What is a primary key?**

  Primary key (PK) is an attribute (or a combination of attributes) that uniquely identifies any given entity (row)

- **What are the criteria for selecting primary key?**

  If you know the value of attribute A, you can look up (determine) the value of attribute B.

- **What is a foreign key?**

  An attribute whose values match primary key values in the related table.

- **What is attribute domain?**

  Each column has a specific range of values known as the **attribute domain.**

- **What is entity integrity?**

Entity integrity is concerned with ensuring that each row of a table has a unique and non-null primary key value

- **What is referential integrity?**

  FK contains a value that refers to an existing valid tuple (row) in another relation.

- **Why is NULL not favorable when inserting data values?**

  No data entry, not permitted in primary key

  - Can represent
    - An unknown attribute value
    - A known, but missing, attribute value
    - A "not applicable" condition

    Can create problems when functions such as COUNT, AVERAGE, and SUM are used

- **Provide 1 example each for the type of relationship below:**
  - **1:1:** each student has only one student ID, and each student ID is assigned to only one person.
  - **1:M:** For example, each customer can have many sales orders.
  - **M:N:** example of a many-to many relationship is one between students and classes. A student can register for many classes, and a class can include many students.

- **Explain how to resolve M:N relationship using the example above:**

  By turning the M:N to two 1:N relationships. o resolve a many-to-many relationship between students and classes, create an additional table called Enrollment with columns EnrollmentID, StudentID, and ClassID.

Chapter 5: Entity Relationship Model

- ER model forms the basis of an ER diagram

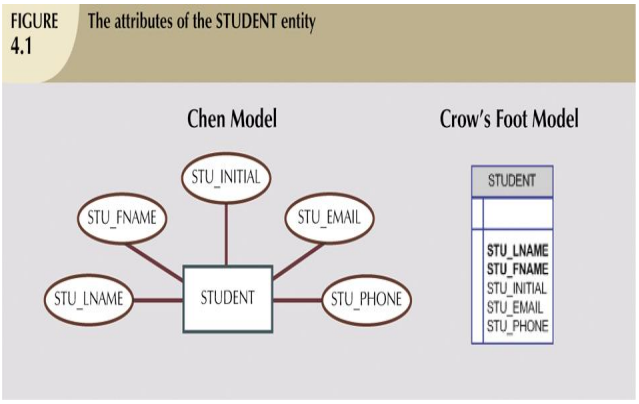
- ERD represents conceptual database as viewed by end user
- ERDs depict database's main components:
  - Entities
  - Attributes
  - Relationships

Entities:

- Refers to entity set and not to single entity occurrence.
- Corresponds to table and not to row in relational environment.
- In both Chen and Crow's Foot models, entity is represented by rectangle containing entity's name.
- Entity name, a noun, is usually written in capital letters.

Attributes:

- Characteristics of entities
- In **Chen model, attributes are represented by ovals** and are connected to entity rectangle with a line.
- Each oval contains the name of attribute it represents.
- In **Crow's Foot model, attributes are written in attribute box below entity rectangle**.



Domains:

- Attributes have domain.
  - **Domain is attribute's set of possible values.**
- Attributes may share a domain.

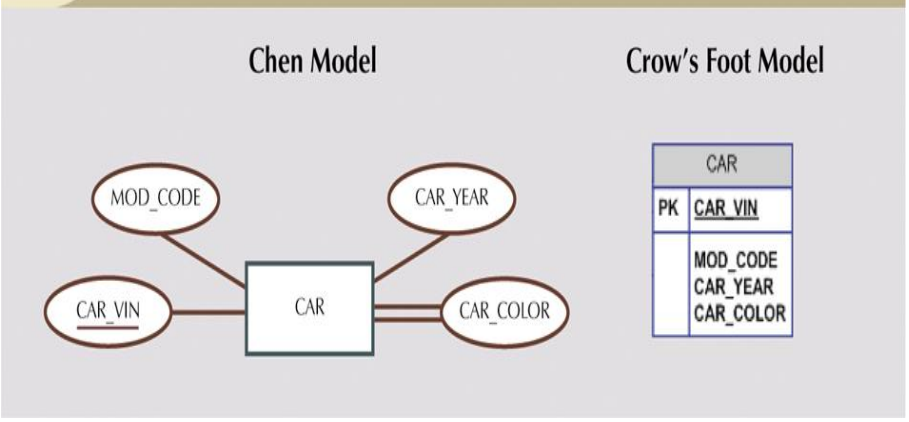
IDENTIFIES FOR PRIMARY KEYS: Underlined in the ERD

Composite Primary Keys:

- Primary keys ideally composed of only single attribute.

- Possible to use a composite key.
  - Primary key composed of more than one attribute.
- Composite attribute can be subdivided.
- Simple attribute cannot be subdivided.
- Single-value attribute can have only a single value.
- Multivalued attributes can have many values

FIGURE 4.3 A multivalued attribute in an entity



Derived Attributes:

- Attribute whose value may be calculated (derived) from other attributes.
- Need not be physically stored within database.
- Can be derived by using an algorithm.

FIGURE 4.6 Depiction of a derived attribute

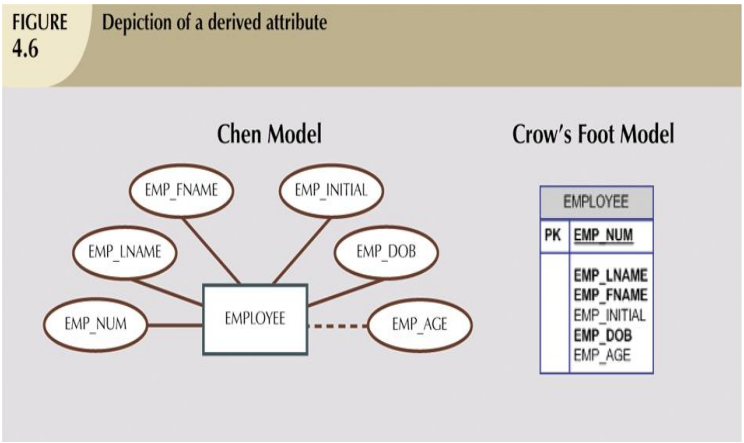


TABLE 4.2 Advantages and Disadvantages of Storing Derived Attributes

	DERIVED ATTRIBUTE	
	STORED	NOT STORED
Advantage	Saves CPU processing cycles Data value is readily available Can be used to keep track of historical data	Saves storage space Computation always yields current value
Disadvantage	Requires constant maintenance to ensure derived value is current, especially if any values used in the calculation change	Uses CPU processing cycles Adds coding complexity to queries

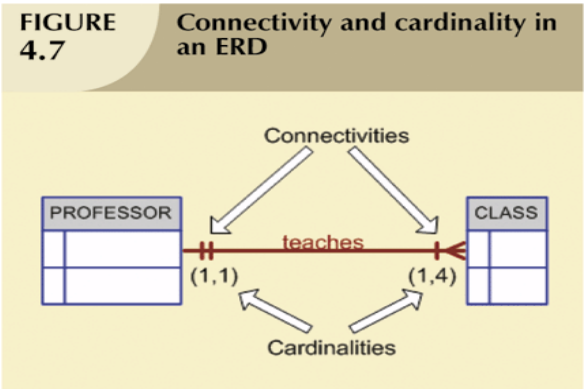
Relationships:

- Association between entities
- Participants are entities that participate in a relationship.
- Relationships between entities always operate in both directions.
- Relationship can be classified as 1:M
- Relationship classification is difficult to establish if know only one side of the relationship.

Connectivity and Cardinality:

- **Connectivity**
  - Used to describe the relationship classification.
- **Cardinality**
  - Expresses minimum and maximum number of entity occurrences associated with one occurrence of related entity.
- Established by very concise statements known as business rules.

FIGURE 4.7 Connectivity and cardinality in an ERD

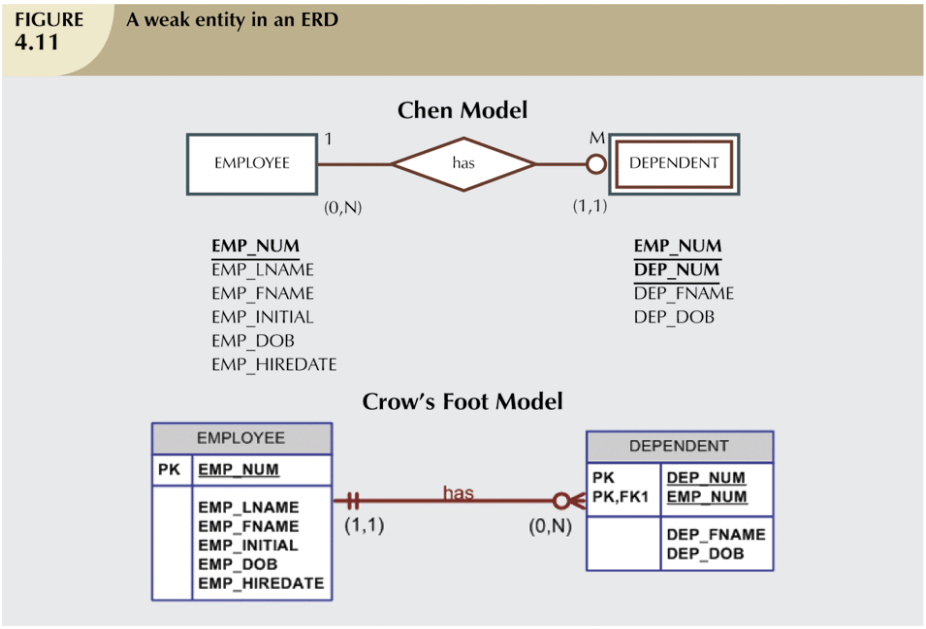


Weak Entities:

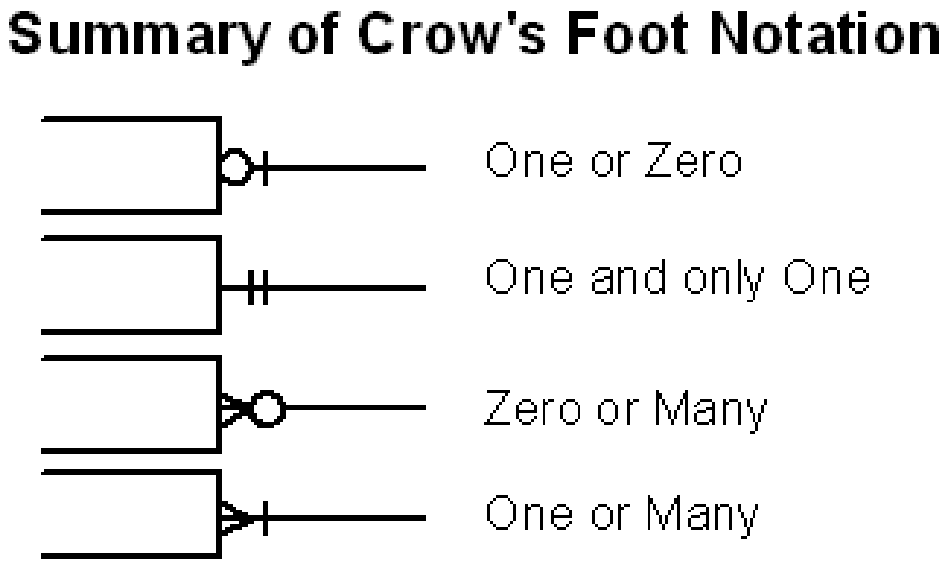
- **Weak entity meets two conditions**

- Existence-dependent
  - Cannot exist without entity with which it has a relationship.
- Has primary key that is partially or totally derived from parent entity in relationship?

- Database designer usually determines whether an entity can be described as weak based on business rules.



Crow's Foot Notations for ERD (IMPORTANT)



Relationship Participation:

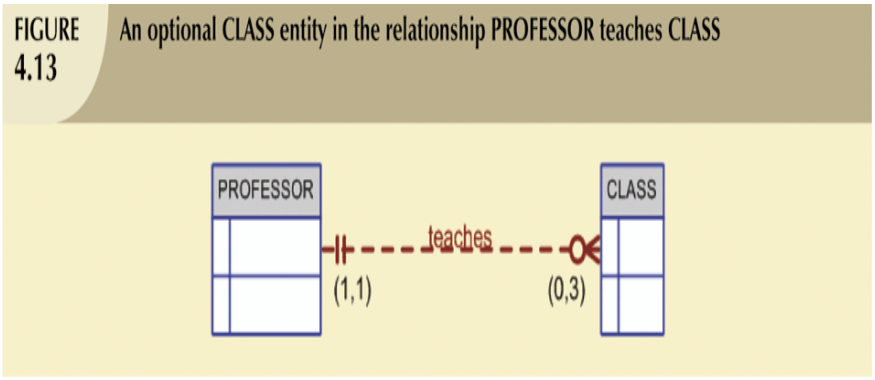
- **Optional participation**

- One entity occurrence does not require corresponding entity occurrence in particular relationship.

- **Mandatory participation**
  - One entity occurrence requires corresponding entity occurrence in particular relationship.

This is how you draw a Optional ERD:

FIGURE 4.13 An optional CLASS entity in the relationship PROFESSOR teaches CLASS



Crow's Foot Symbols (IMPORTANT)

TABLE 4.3 Crow's Foot Symbols

CROW'S FOOT SYMBOL	CARDINALITY	COMMENT
	(0,N)	Zero or many. Many side is optional.
	(1,N)	One or many. Many side is mandatory.
	(1,1)	One and only one. 1 side is mandatory.
	(0,1)	Zero or one. 1 side is optional.

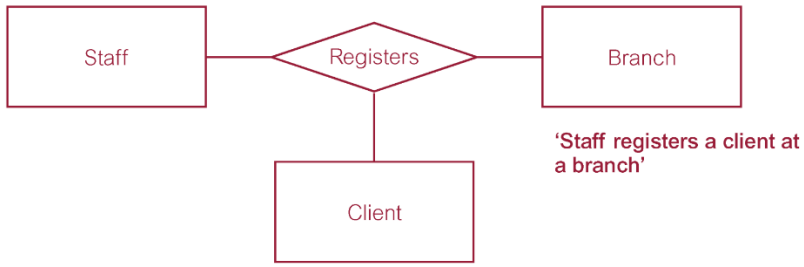
Relationship Degree:

- **Indicates number of entities or participants associated with a relationship**
- **Unary relationship**
  - Association is maintained within single entity .
- **Binary relationship**
  - Two entities are associated.
- **Ternary relationship**
  - Three entities are associated.
- **Quaternary relationship**

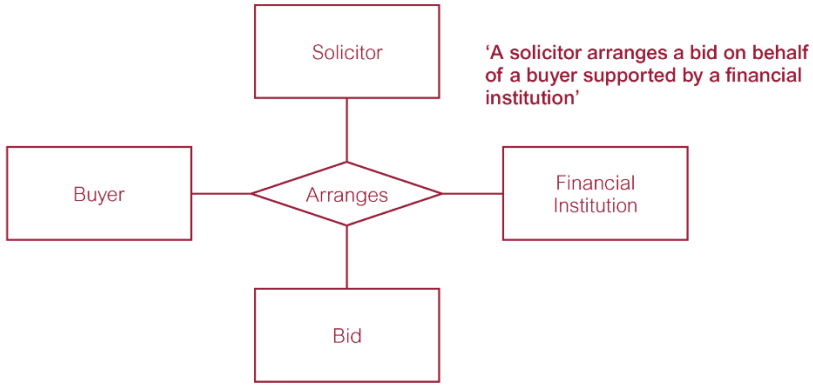
Four entities are associated.

Example of Binary Relationship is POwns:

Example of Ternary Relationship called *Registers:*



Example of Quaternary Relationship called *Arranges:*



Recursive Relationships:

- Relationship can exist between occurrences of the same entity set
- Naturally found within unary relationship

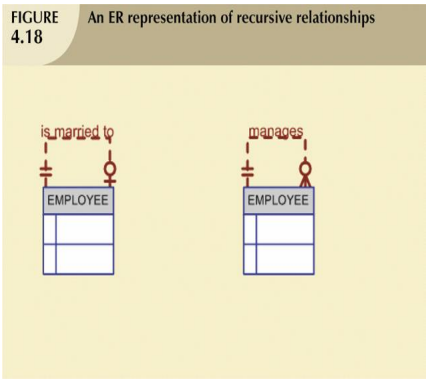


FIGURE 4.19 The 1:1 recursive relationship "EMPLOYEE is married to EMPLOYEE"

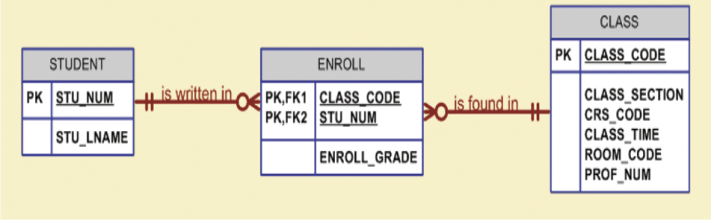
Database name: CH04_PartCo
Table name: EMPLOYEE_V1

	EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_SPOUSE
▶	345	Ramirez	James	347
	346	Jones	Anne	349
	347	Ramirez	Louise	345
	348	Delaney	Robert	
	349	Shapiro	Anton	346

Composite Entities:

- Also known as bridge entities
- Composed of primary keys of each of the entities to be connected
- May also contain additional attributes that play no role in connective process

FIGURE 4.26 A composite entity in an ERD



Chapter 5 Questions:

- **Provide 1 example each for the relationship degree below.**
  - **Unary / recursive**

    Example: Employee and Manager Relationship

    Explanation: In an organization, an employee can also be a manage
  - **Binary**

    Example: Student and Course Relationship

    Explanation: A student enrols in multiple courses, and each course is taken by multiple students.
  - **Ternary**

    Example: Doctor, Patient, and Appointment Relationship

    Explanation: In a healthcare system, a doctor may have multiple patients, and each patient may have multiple appointments with different doctors.
  - **Quaternary**

    Example: Car, Driver, Insurance Company, and Accident Relationship

Explanation: In the context of car accidents, there is a quaternary relationship involving cars, drivers, insurance companies, and accidents.

[redacted]

Chapter 6: Normalization

Normalization

- **Process for evaluating and correcting table structures to minimize data redundancies.**
  - Reduces data anomalies.
- **Works through a series of stages called normal forms:**
  - First normal form (1NF)
  - Second normal form (2NF)
  - Third normal form (3NF)
- 2NF is better than 1NF; 3NF is better than 2NF
- For most business database design purposes, 3NF is as high as we need to go in normalization process
- Highest level of normalization is not always most desirable.

Why is Normalization Needed?

- **Example: Company that manages building projects**
  - **Charges its clients by billing hours spent on each contract.**
  - **Hourly billing rate is dependent on employee's position.**
  - **Periodically, report is generated that contains information displayed in Table 5.1**

TABLE 5.1 A Sample Report Layout

PROJ. NUM.	PROJECT NAME	EMPLOYEE NUMBER	EMPLOYEE NAME	JOB CLASS.	CHG/ HOUR	HOURS BILLED	TOTAL CHARGE
15	Evergreen	103	June E. Arbough	Elec. Engineer	$ 85.50	23.8	$ 2,011.10
		101	John G. News	Database Designer	$105.00	19.4	$ 2,037.00
		105	Alice K. Johnson*	Database Designer	$105.00	35.7	$ 3,748.50
		106	William Smithfield	Programmer	$ 35.75	12.6	$ 450.45
		102	David H. Senior	Systems Analyst	$ 96.75	23.8	$ 2,302.65
				Subtotal			**$10,549.70**
18	Amber Wave	114	Annelise Jones	Applications Designer	$ 48.10	25.6	$ 1,183.26
		118	James J. Frommer	General Support	$ 18.36	45.3	$ 831.71
		104	Anne K. Ramoras*	Systems Analyst	$ 96.75	32.4	$ 3,135.70
		112	Darlene M. Smithson	DSS Analyst	$ 45.95	45.0	$ 2,021.80
				Subtotal			**$ 7,172.47**
22	Rolling Tide	105	Alice K. Johnson	Database Designer	$105.00	65.7	$ 6,793.50
		104	Anne K. Ramoras	Systems Analyst	$ 96.75	48.4	$ 4,682.70
		113	Delbert K. Joenbrood*	Applications Designer	$ 48.10	23.6	$ 1,135.16
		111	Geoff B. Wabash	Clerical Support	$ 26.87	22.0	$ 591.14
		106	William Smithfield	Programmer	$ 35.75	12.8	$ 457.60
				Subtotal			**$13,660.10**
25	Starflight	107	Maria D. Alonzo	Programmer	$ 35.75	25.6	$ 879.45
		115	Travis B. Bawangi	Systems Analyst	$ 96.75	45.8	$ 4,431.15
		101	John G. News*	Database Designer	$105.00	56.3	$ 5,911.50
		114	Annelise Jones	Applications Designer	$ 48.10	33.1	$ 1,592.11
		108	Ralph B. Washington	Systems Analyst	$ 96.75	23.6	$ 2,283.30
		118	James J. Frommer	General Support	$ 18.36	30.5	$ 559.98
		112	Darlene M. Smithson	DSS Analyst	$ 45.95	41.4	$ 1,902.33
				Subtotal			**$17,559.82**
				Total			**$48,942.09**

Note: * indicates project leader.

- **Structure of data set in Figure 5.1 does not handle data very well.**
- **The table structure appears to work; report generated with ease.**
- **Unfortunately, report may yield different results depending on what data anomaly has occurred.**
- **Each table represents a single subject.**
- **No data item will be unnecessarily stored in more than one table.**
- **All attributes in a table are dependent on the primary key.**

TABLE 5.2 Normal Forms

NORMAL FORM	CHARACTERISTIC	SECTION
First normal form (1NF)	Table format; no repeating groups and PK identified	5.3.1
Second normal form (2NF)	1NF and no partial dependencies	5.3.2
Third normal form (3NF)	2NF and no transitive dependencies	5.3.3
Boyce-Codd normal form (BCNF)	Every determinant is a candidate key (special case of 3NF)	5.6.1
Fourth normal form (4NF)	3NF and no independent multivalued dependencies	5.6.2

Conversion For 1NF (First Normal Form)

Repeating group

- Derives its name from the fact that a group of multiple entries of same type can exist for any single key attribute occurrence.

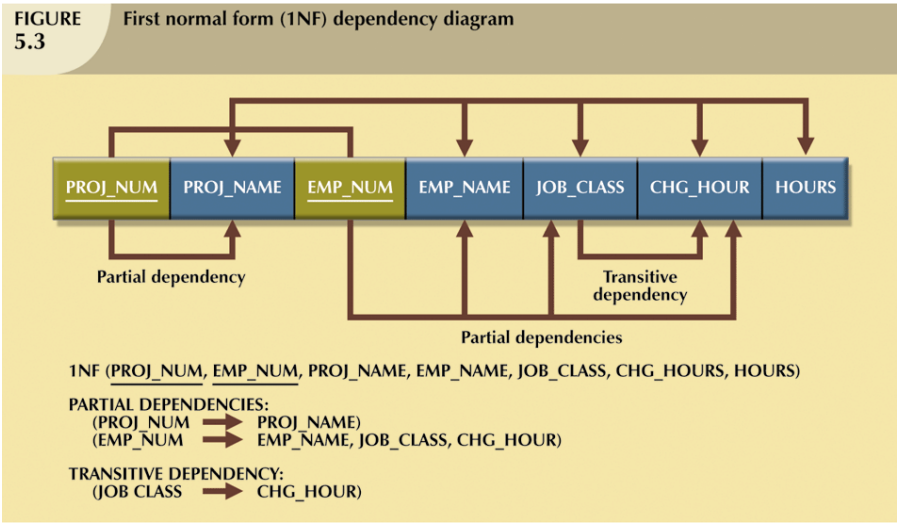
Relational table must not contain repeating groups.

Normalizing table structure will reduce data redundancies.

Normalization is three-step procedure.

- **Step 1: Eliminate the Repeating Groups**
  - Present data in tabular format, where each cell has single value and there are no repeating groups.
  - Eliminate repeating groups, eliminate nulls by making sure that each repeating group attribute contains an appropriate data value.
- **Step 2: Identify the Primary Key**
  - Primary key must uniquely identify attribute value.
  - New key must be composed.

- **Step 3: Identify All Dependencies**
  - Dependencies can be depicted with help of a diagram.
  - Dependency diagram:
    - Depicts all dependencies found within given table structure.
    - Helpful in getting bird's-eye view of all relationships among table's attributes.
    - Makes it less likely that will overlook an important dependency.



First normal form describes tabular format in which:

- All key attributes are defined.
- There are no repeating groups in the table.
- All attributes are dependent on primary key.

All relational tables satisfy 1NF requirements.

Some tables contain partial dependencies.

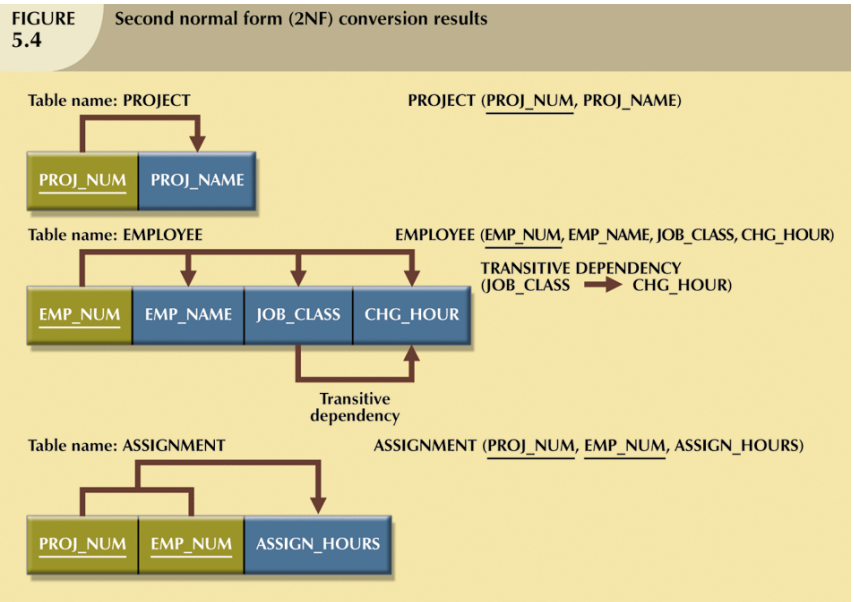
- Dependencies based on only part of the primary key.
- Sometimes used for performance reasons but should be used with caution.

Still subject to data redundancies

Conversion For 2NF (Second Normal Form)

Relational database design can be improved by converting the database into second normal form (2NF). Two steps.

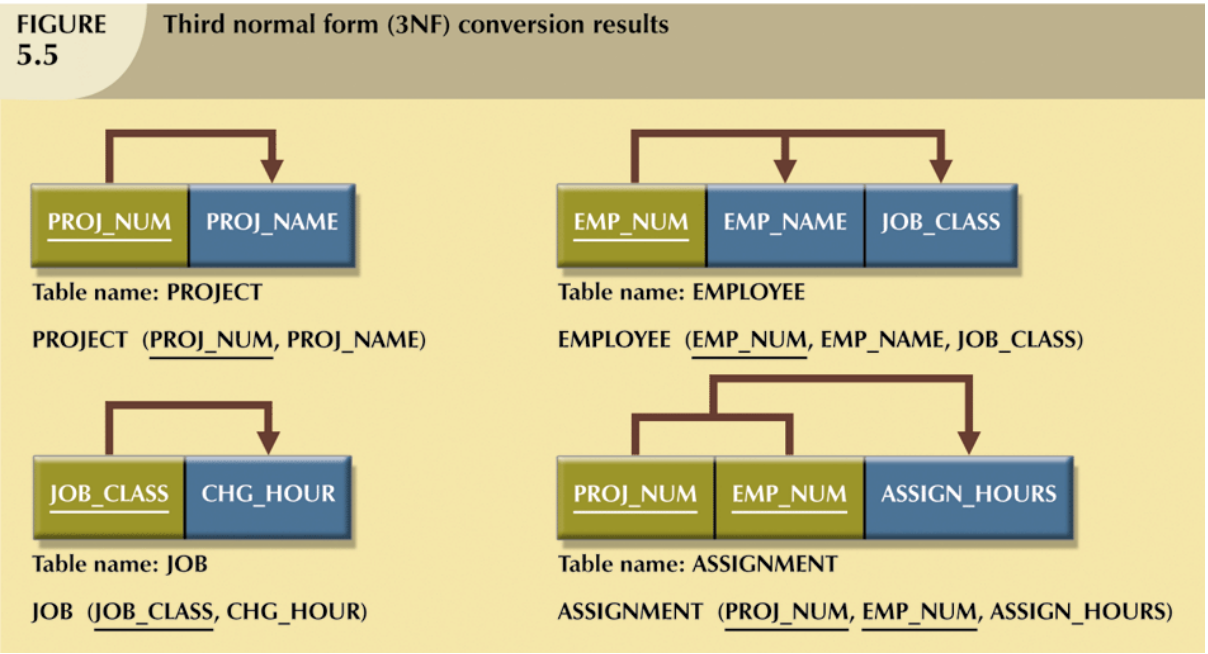
- **Step 1: Write Each Key Component on a Separate Line**
  - **Write each key component on separate line, then write original (composite) key on last line**
  - **Each component will become key in new table**
- **Step 2: Assign Corresponding Dependent Attributes**
  - **Determine those attributes that are dependent on other attributes**
  - **At this point, most anomalies have been eliminated**



- **Table is in second normal form (2NF) when:**
  - **It is in 1NF and**
  - **It includes no partial dependencies:**
    - **No attribute is dependent on only portion of primary key**

Conversion For 3NF (Third Normal Form)

- **Data anomalies created are easily eliminated by completing three steps**
- **Step 1: Identify Each New Determinant**
  - **For every transitive dependency, write its determinant as PK for new table**
    - **Determinant :Any attribute whose value determines other values within a row**
- **Step 2: Identify the Dependent Attributes**
  - **Identify attributes dependent on each determinant identified in Step 1 and identify dependency**
  - **Name table to reflect its contents and function**
- **Step 3: Remove the Dependent Attributes from Transitive Dependencies**
  - **Eliminate all dependent attributes in transitive relationship(s) from each of the tables that have such a transitive relationship**
  - **Draw new dependency diagram to show all tables defined in Steps 1–3**
  - **Check new tables as well as tables modified in Step 3 to make sure that each table has determinant and that no table contains inappropriate dependencies**



- **A table is in third normal form (3NF) when both of the following are true:**
  - **It is in 2NF**
  - **It contains no transitive dependencies**

Questions for TOPIC 6:

- **What is partial dependency?**

  Partial dependency occurs in a relational database when a non-prime attribute (an attribute that is not part of any candidate key) is functionally dependent on only a part (proper subset) of a candidate key, rather than the entire key.

- **What is transitive dependency?**
  Transitive dependency occurs when an attribute is functionally dependent on another non-prime attribute, which itself is dependent on the primary key.

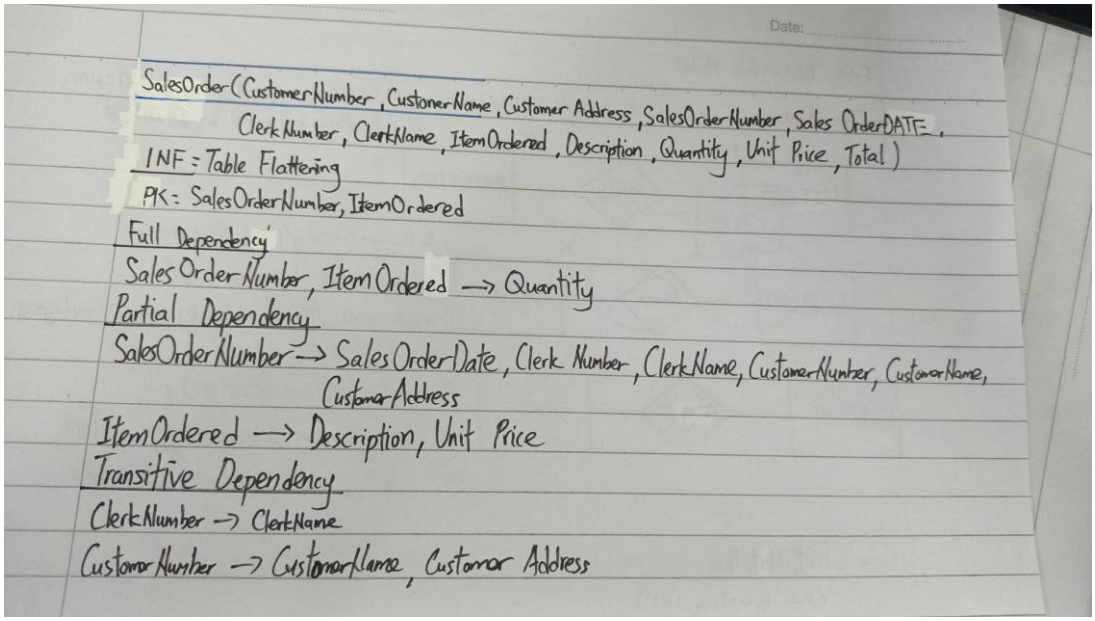
Given the un-normalized form, normalized it into 3NF

Sales Order

Fiction Company
202 N. Main
Mahattan, KS 66502

CustomerNumber:	*1001*	**Sales Order Number:**	405
Customer Name:	*ABC Company*	**Sales Order Date:**	2/1/2000
Customer Address:	*100 Points*	**Clerk Number:**	*210*
	Manhattan, KS 66502	**Clerk Name:**	*Martin Lawrence*

Item Ordered	Description	Quantity	Unit Price	Total
800	*widgit small*	*40*	*60.00*	*2,400.00*
801	*tingimajigger*	*20*	*20.00*	*400.00*
805	*thingibob*	*10*	*100.00*	*1,000.00*
Order Total				*3,800.00*



Chapter 7: SQL

- **SQL functions fit into two broad categories:**
  - **Data definition language (DDL)**
    - SQL includes commands to:
      - Create database objects, such as tables, indexes, and views.
      - Define access rights to those database objects.
  - **Data manipulation language (DML)**
    - Includes commands to insert, update, delete, and retrieve data within database tables.

- American National Standards Institute (ANSI) prescribes a standard SQL

TABLE 7.1 SQL Data Definition Commands

COMMAND OR OPTION	DESCRIPTION
CREATE SCHEMA AUTHORIZATION	Creates a database schema
CREATE TABLE	Creates a new table in the user's database schema
NOT NULL	Ensures that a column will not have null values
UNIQUE	Ensures that a column will not have duplicate values
PRIMARY KEY	Defines a primary key for a table
FOREIGN KEY	Defines a foreign key for a table
DEFAULT	Defines a default value for a column (when no value is given)
CHECK	Constraint used to validate data in an attribute
CREATE INDEX	Creates an index for a table
CREATE VIEW	Creates a dynamic subset of rows/columns from one or more tables
ALTER TABLE	Modifies a table's definition (adds, modifies, or deletes attributes or constraints)
CREATE TABLE AS	Creates a new table based on a query in the user's database schema
DROP TABLE	Permanently deletes a table (and thus its data)
DROP INDEX	Permanently deletes an index
DROP VIEW	Permanently deletes a view

TABLE 7.2 SQL Data Manipulation Commands

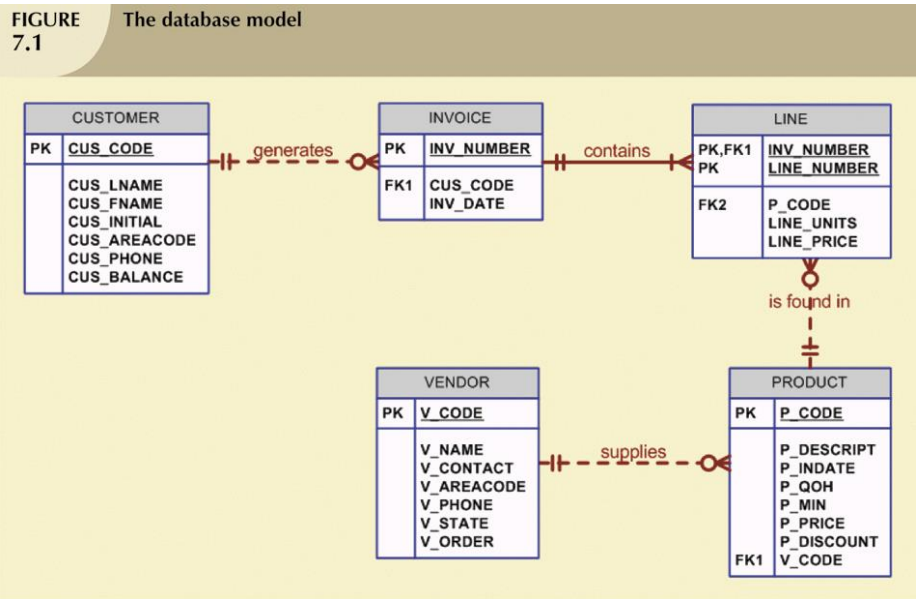
COMMAND OR OPTION	DESCRIPTION
INSERT	Inserts row(s) into a table
SELECT	Selects attributes from rows in one or more tables or views
WHERE	Restricts the selection of rows based on a conditional expression
GROUP BY	Groups the selected rows based on one or more attributes
HAVING	Restricts the selection of grouped rows based on a condition
ORDER BY	Orders the selected rows based on one or more attributes
UPDATE	Modifies an attribute's values in one or more table's rows
DELETE	Deletes one or more rows from a table
COMMIT	Permanently saves data changes
ROLLBACK	Restores data to their original values

TABLE 7.2 SQL Data Manipulation Commands (continued)

COMMAND OR OPTION	DESCRIPTION
COMPARISON OPERATORS	
=, <, >, <=, >=, <>	Used in conditional expressions
LOGICAL OPERATORS	
AND/OR/NOT	Used in conditional expressions
SPECIAL OPERATORS	Used in conditional expressions
BETWEEN	Checks whether an attribute value is within a range
IS NULL	Checks whether an attribute value is null
LIKE	Checks whether an attribute value matches a given string pattern
IN	Checks whether an attribute value matches any value within a value list
EXISTS	Checks whether a subquery returns any rows
DISTINCT	Limits values to unique values
AGGREGATE FUNCTIONS	Used with SELECT to return mathematical summaries on columns
COUNT	Returns the number of rows with non-null values for a given column
MIN	Returns the minimum attribute value found in a given column
MAX	Returns the maximum attribute value found in a given column
SUM	Returns the sum of all values for a given column
AVG	Returns the average of all values for a given column

Database Model:

FIGURE 7.1 The database model



- **Authentication**
  - **Process through which DBMS verifies that only registered users are able to access database**
  - **Log on to RDBMS using user ID and password created by database administrator**
- **Schema**
  - **Group of database objects—such as tables and indexes—that are related to each other**

- **NOT NULL constraint**
  - **Ensures that column does not accept nulls**
- **UNIQUE constraint**
  - **Ensures that all values in column are unique**
- **DEFAULT constraint**
  - **Assigns value to attribute when a new row is added to table**
- **CHECK constraint**
  - **Validates data when attribute value is entered**
- **INSERT**
  - **Used to enter data into table**
  - **Syntax:**
    - **INSERT INTO *columnname***
      VALUES (*value1, value2, … , valuen*);
- **Changes made to table contents are not physically saved on disk until, one of the following occurs:**
  - **Database is closed**
  - **Program is closed**
  - **COMMIT command is used**
- **Syntax:**
  - **COMMIT [WORK];**

Will permanently save any changes made to any table in the database

- **SELECT**
  - **Used to list contents of table**
  - **Syntax:**
    - **SELECT *columnlist***
      FROM *tablename*;
- **Asterisk can be used as wildcard character to list all attributes**
- **UPDATE**
  - **Modify data in a table**
  - **Syntax:**

        - **UPDATE *tablename***
          SET *columnname* = *expression* [, *columname* = *expression*]
          [WHERE *conditionlist*];

- **ROLLBACK**
    - **Used to restore database to its previous condition**
    - **Only applicable if COMMIT command has not been used to permanently store changes in database**
- **Syntax:**
    - **ROLLBACK;**
- **COMMIT and ROLLBACK only work with data manipulation commands that are used to add, modify, or delete table rows**
- **DELETE**
    - **Deletes a table row**
    - **Syntax:**
        - **DELETE FROM *tablename***
          [WHERE *conditionlist*];
- **WHERE condition is optional**
- **If WHERE condition is not specified, all rows from specified table will be deleted**
- **INSERT**
    - **Inserts multiple rows from another table (source)**
    - **Uses SELECT subquery**
        - **Query that is embedded (or nested) inside another query**
        - **Executed first**
    - **Syntax:**
        - **INSERT INTO *tablename* SELECT *columnlist* FROM *tablename*;**
- **BETWEEN**
    - **Used to check whether attribute value is within a range**
- **IS NULL**
    - **Used to check whether attribute value is null**
- **LIKE**
    - **Used to check whether attribute value matches given string pattern**
- **IN**
    - **Used to check whether attribute value matches any value within a value list**

- **EXISTS**
  - **Used to check if subquery returns any rows**
- **All changes in table structure are made by using ALTER command**
  - **Followed by keyword that produces specific change**
  - **Following three options are available:**
    - **ADD**
    - **MODIFY**
    - **DROP**
- **When table is copied, integrity rules do not copy, so primary and foreign keys need to be manually defined on new table**
- **User ALTER TABLE command**
  - **Syntax:**
    - **ALTER TABLE *tablename* ADD PRIMARY KEY(*fieldname*);**
    - **For foreign key, use FOREIGN KEY in place of PRIMARY KEY**
- **DROP**
  - **Deletes table from database**
  - **Syntax:**
    - **DROP TABLE *tablename*;**

Inner Join:

Returns only the rows where there is a match in both tables based on the specified condition.

Rows from the tables that do not satisfy the condition are excluded from the result set.

Example:

SELECT * FROM Table1 INNER JOIN Table2 ON Table1.ID = Table2.ID;

Left Join (or Left Outer Join):

Returns all rows from the left table and the matched rows from the right table.

If there is no match in the right table, NULL values are returned for columns from the right table.

Example:

SELECT * FROM Table1 LEFT JOIN Table2 ON Table1.ID = Table2.ID;

Right Join (or Right Outer Join):

Returns all rows from the right table and the matched rows from the left table.

If there is no match in the left table, NULL values are returned for columns from the left table.

Example:

SELECT * FROM Table1 RIGHT JOIN Table2 ON Table1.ID = Table2.ID;

Full Join (or Full Outer Join):

Returns all rows when there is a match in either the left or right table.

If there is no match in one of the tables, NULL values are returned for columns from the table without a match.

Example:

SELECT * FROM Table1 FULL JOIN Table2 ON Table1.ID = Table2.ID;

Inner Join returns matched rows.

Left Join returns all rows from the left table and matched rows from the right table.

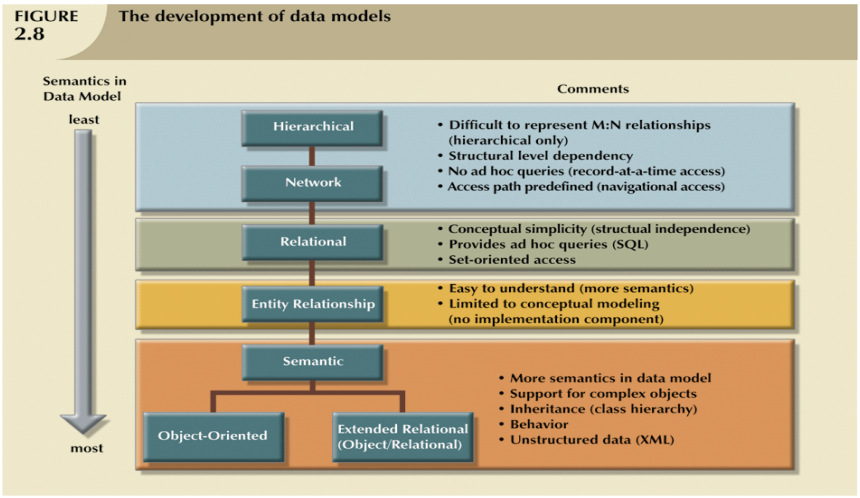
Right Join returns all rows from the right table and matched rows from the left table.

Full Join returns all rows when there is a match in either the left or right table.

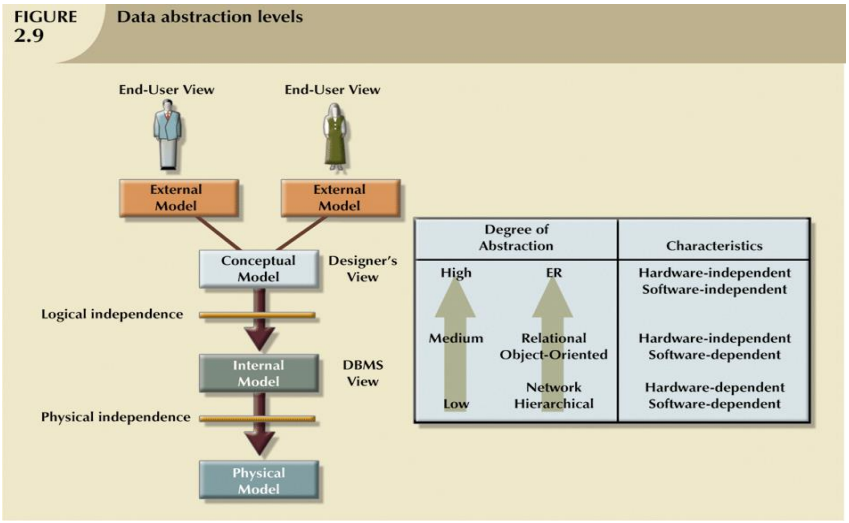
Questions for Topic 7:

- **Describe the difference between DDL & DML**
  - **Data definition language (DDL)**
    - SQL includes commands to:
      - Create database objects, such as tables, indexes, and views.
      - Define access rights to those database objects.
  - **Data manipulation language (DML)**
    - Includes commands to insert, update, delete, and retrieve data within database tables.

Chapter 8: Extended Entity Relationship Model

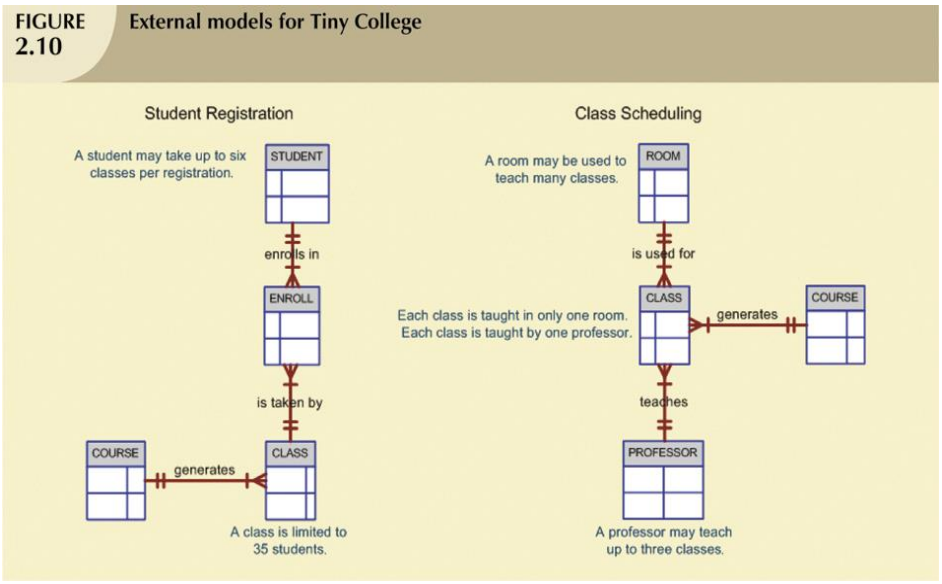


- **American National Standards Institute (ANSI) Standards Planning and Requirements Committee (SPARC)**
  - Defined a framework for data modeling based on degrees of data abstraction(1970s):
    - External
    - Conceptual
    - Internal



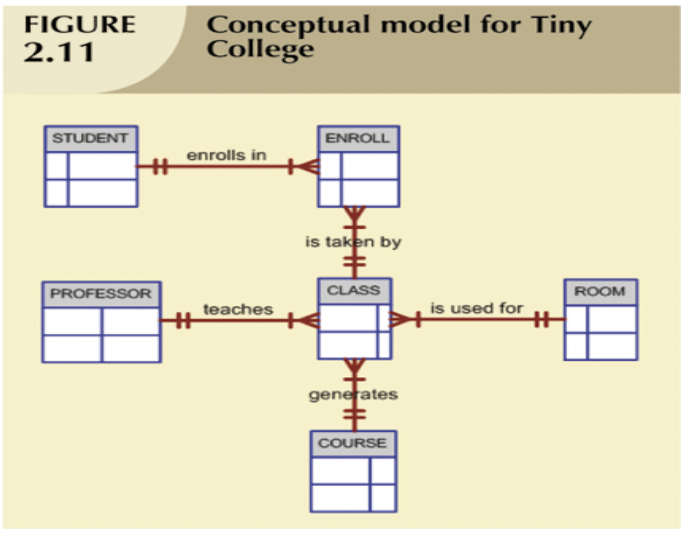
The External Model: End users' view of the data environment

- **Advantages:**
  - Easy to identify specific data required to support each business unit's operations.
  - Facilitates designer's job by providing feedback about the model's adequacy.
  - Creation of external models helps to ensure security constraints in the database design.
  - Simplifies application program development.



The Conceptual Model:

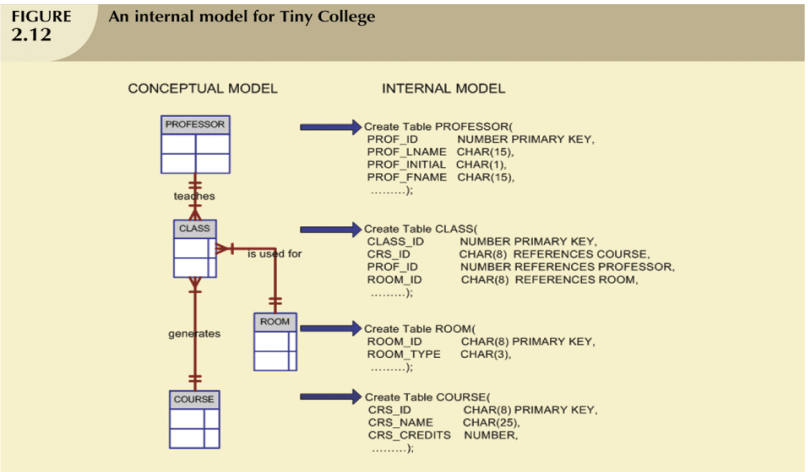
- Represents global view of the entire database
- Basis for identification and high-level description of main data objects, avoiding details
- Most widely used conceptual model is the entity relationship (ER) model



- Provides a relatively easily understood macro level view of data environment
- Independent of both software and hardware
  - Does not depend on the DBMS software used to implement the model
  - Does not depend on the hardware used in the implementation of the model

The Internal Model:

- **Representation of the database as "seen" by the DBMS**
- **Maps the conceptual model to the DBMS**
- **Internal schema depicts a specific representation of an internal model**



The Physical Model:

- **Operates at lowest level of abstraction, describing the way data are saved on storage media such as disks or tapes**
- **Software and hardware dependent**

TABLE 2.3 Levels of Data Abstraction

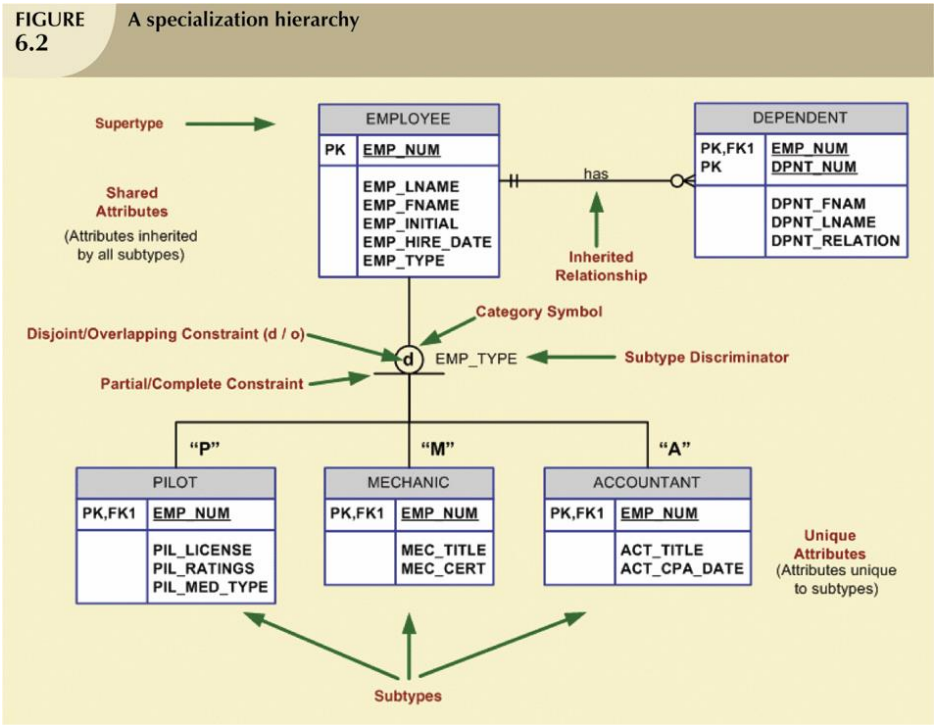
MODEL	DEGREE OF ABSTRACTION	FOCUS	INDEPENDENT OF
External	High	End-user views	Hardware and software
Conceptual	↕	Global view of data (independent of database model)	Hardware and software
Internal		Specific database model	Hardware
Physical	Low	Storage and access methods	Neither hardware nor software

Entity Supertypes and Subtypes

- **Entity supertype**
  - Generic entity type that is related to one or more entity subtypes
  - Contains common characteristics
- **Entity subtypes**
  - Contains unique characteristics of each entity subtype

Specialization Hierarchy:

- **Depicts arrangement of higher-level entity supertypes (parent entities) and lower-level entity subtypes (child entities)**
- **Relationships sometimes described in terms of "IS-A" relationships**



- Support attribute inheritance
- Define special supertype attribute known as subtype discriminator
- Define disjoint/overlapping constraints and complete/partial constraints

Inheritance:

- Enables entity subtype to inherit attributes and relationships of supertype
- All entity subtypes inherit their primary key attribute from their supertype
- At implementation level, supertype and its subtype(s) depicted in specialization hierarchy maintain a 1:1 relationship

FIGURE 6.3 The EMPLOYEE-PILOT supertype-subtype relationship

Table Name: EMPLOYEE

EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIRE_DATE	EMP_TYPE
100	Kolmycz	Xavier	T	15-Mar-88	
101	Lewis	Marcos		25-Apr-89	P
102	Vandam	Jean		20-Dec-93	A
103	Jones	Victoria	R	28-Aug-03	
104	Lange	Edith		20-Oct-97	P
105	Williams	Gabriel	U	08-Nov-97	P
106	Duzak	Mario		05-Jan-04	P
107	Diante	Venite	L	02-Jul-97	M
108	Wiesenbach	Joni		18-Nov-95	M
109	Travis	Brett	T	14-Apr-01	P
110	Genkazi	Stan		01-Dec-03	A

Table Name: PILOT

EMP_NUM	PIL_LICENSE	PIL_RATINGS	PIL_MED_TYPE
101	ATP	SEL/MEL/Instr/CFII	1
104	ATP	SEL/MEL/Instr	1
105	COM	SEL/MEL/Instr/CFI	2
106	COM	SEL/MEL/Instr	2
109	COM	SEL/MEL/SES/Instr/CFII	1

Subtype Discriminator: The attribute in supertype entity that determines to which entity subtype each supertype occurrence is related

Disjoint subtypes

- Also known as non-overlapping subtypes
- Subtypes that contain unique subset of supertype entity set

Overlapping subtypes

Subtypes that contain nonunique subsets of supertype entity set

Completeness Constraint:

- Specifies whether each entity supertype occurrence must also be member of at least one subtype
- Can be partial or total

Specialization and Generalization:

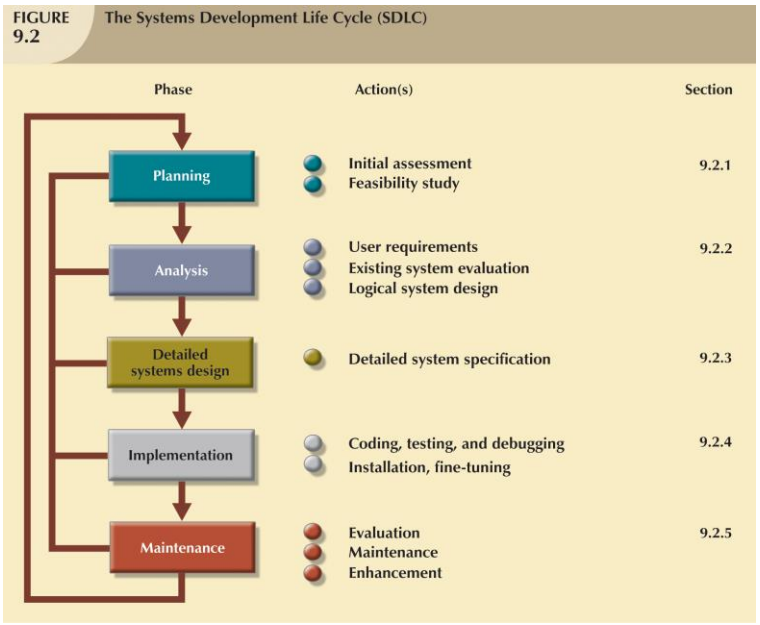
- **Specialization**
  - Top-down process of identifying lower-level, more specific entity subtypes from higher-level entity supertype
  - Based on grouping unique characteristics and relationships of the subtypes
- **Generalization**
  - Bottom-up process of identifying higher-level, more generic entity supertype from lower-level entity subtypes
  - Based on grouping common characteristics and relationships of the subtypes

Chapter 9: Database Dev Life Cycle

The Information System:

- Provides for data collection, storage, and retrieval
- Composed of people, hardware, software, database(s), application programs, and procedures
- **Systems analysis**
  - Process that establishes need for and extent of information system
- **Systems development**
  - Process of creating information system
- **Applications**
  - Transform data into information that forms basis for decision making
  - **Usually produce the following:**
    - Formal report
    - Tabulations
    - Graphic displays
  - **Composed of following two parts:**
    - Data
    - Code by which data are transformed into information
- **Information system performance depends on triad of factors:**
  - Database design and implementation
  - Application design and implementation
  - Administrative procedures
- **Database development**
  - Process of database design and implementation
  - Primary objective is to create complete, normalized, nonredundant (to the extent possible), and fully integrated conceptual, logical, and physical database models

The Systems Development Life Cycle (SDLC):



- Traces history (life cycle) of information system
- Provides "big picture" within which database design and application development can be mapped out and evaluated

Divided into following five phases:

Planning

- Yields general overview of company and its objectives.
- Initial assessment made of information-flow-and-extent requirements.
- Must begin to study and evaluate alternate solutions.
  - Technical aspects of hardware and software requirements
  - System cost

Analysis

- Problems defined during planning phase are examined in greater detail during analysis
- Thorough audit of user requirements
- Existing hardware and software systems are studied.
- Goal is better understanding of system's functional areas, actual and potential problems, and opportunities.
- Includes creation of logical system design
- Must specify appropriate conceptual data model, inputs, processes, and expected output requirements

- Might use tools such as data flow diagrams (DFDs), hierarchical input process output (HIPO) diagrams, and entity relationship (ER) diagrams
- Yields functional descriptions of system's components (modules) for each process within database environment

Detailed systems design

- Designer completes design of system's processes.
- Includes all necessary technical specifications.
- Steps are laid out for conversion from old to new system.
- Training principles and methodologies are also planned.

Implementation

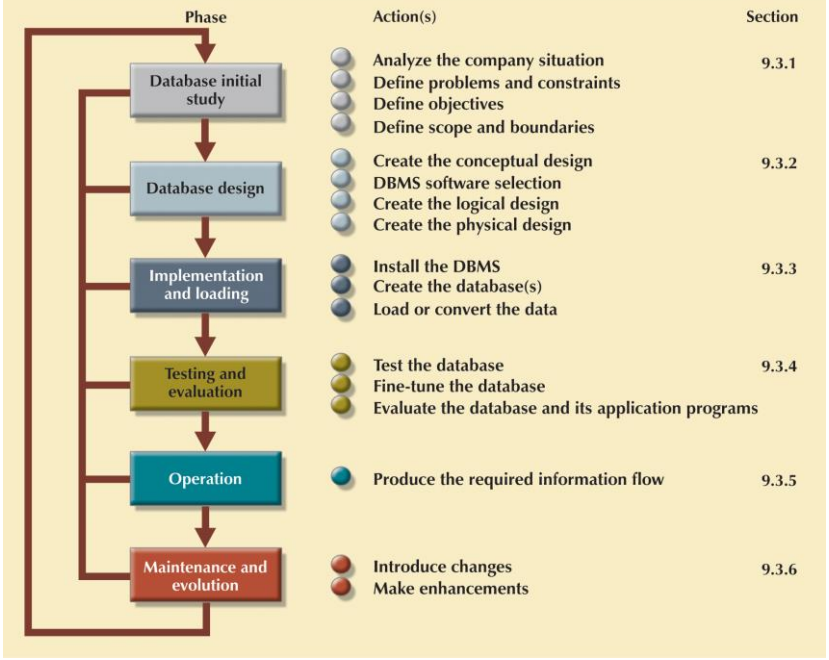
- Hardware, DBMS software, and application programs are installed, and database design is implemented
- Cycle of coding, testing, and debugging continues until database is ready to be delivered
- Database is created and system is customized by creation of tables and views, and user authorizations

Maintenance

- Maintenance activities group into three types:
  - Corrective maintenance in response to systems errors
  - Adaptive maintenance due to changes in business environment
  - Perfective maintenance to enhance system
- Computer-assisted systems engineering
  - Make it possible to produce better systems within reasonable amount of time and at reasonable cost

Iterative rather than sequential process

FIGURE 9.3 The Database Life Cycle (DBLC)



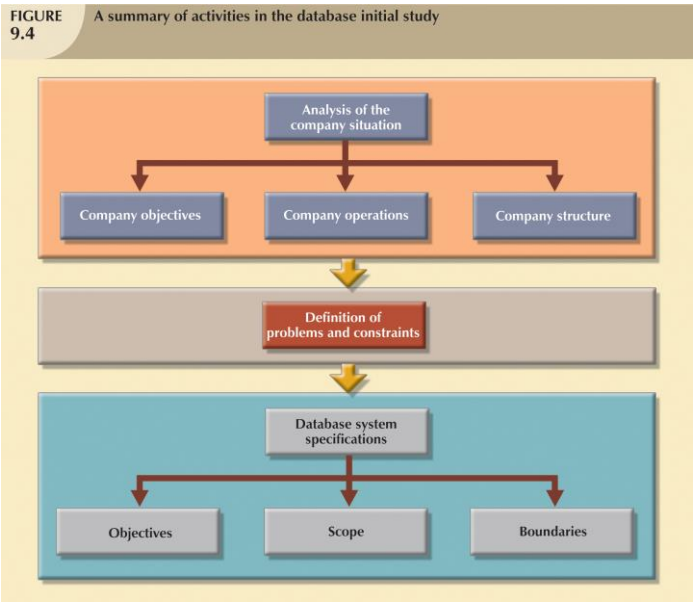
Overall purpose:

- Analyze company situation
- Define problems and constraints
- Define objectives
- Define scope and boundaries

Interactive and iterative processes required to complete first phase of DBLC successfully.

The Database Initial Study:

FIGURE 9.4 A summary of activities in the database initial study



Analyze the Company Situation:

- Analysis–To break up any whole into its parts so as to find out their nature, function, and so on
- Company situation
  - General conditions in which company operates, its organizational structure, and its mission
- Analyze company situation
  - Discover what company's operational components are, how they function, and how they interact

Define Problems and Constraints:

- Managerial view of company's operation is often different from that of end users
- Designer must continue to carefully probe to generate additional information that will help define problems within larger framework of company operations
- Finding precise answers is important
- Defining problems does not always lead to perfect solution

Define Objectives:

- Designer must ensure that database system objectives correspond to those envisioned by end user(s)
- Designer must begin to address following questions:
  - What is proposed system's initial objective?
  - Will system interface with other existing or future systems in the company?
  - Will system share data with other systems or users?

Define Scope and Boundaries:

- Scope
  - Defines extent of design according to operational requirements
  - Helps define required data structures, type and number of entities, and physical size of database
- Boundaries
  - Limits external to system
  - Often imposed by existing hardware and software

QUESTIONS THAT MAY COME OUT + ANSWERS (MEMORISE THIS SHIT DAWG)

1)

Staff (Staff_ID, Name, Gender, Address, MonthlySalary, Dept_ID)
Department (Dept_ID, Name, Phone_Extension)

Based on the relations above, write structured query language (SQL) for the queries given below.

a) List all male staff who earn below 4000 from 'Marketing' department

```
SELECT
    Staff.Staff_ID,
    Staff.Name,
    Staff.Gender,
    Staff.Address,
    Staff.MonthlySalary,
    Department.Name AS DepartmentName
FROM
    Staff
JOIN
    Department ON Staff.Dept_ID = Department.Dept_ID
WHERE
    Staff.Gender = 'Male'
    AND Staff.MonthlySalary < 4000
    AND Department.Name = 'Marketing';
```

(3 marks)

b) Find the total number of staff from each department

```
SELECT
    Department.Name AS DepartmentName,
    COUNT(*) AS TotalStaff
FROM
    Staff
JOIN
    Department ON Staff.Dept_ID = Department.Dept_ID
GROUP BY
    Department.Name;
```

(3 marks)

c) What is the average annual salary of a staff in the company?

```
SELECT
    AVG(MonthlySalary) AS AverageAnnualSalary
FROM
    Staff;
```

(3 marks)

d) Add a new department named Sales (D04), phone extension is 5014

```
INSERT INTO Department (Dept_ID, Name, Phone_Extension)
VALUES ('D04', 'Sales', '5014');
```

(3 marks)

e) The staff have been given 10% of increment in salary, modify this in the record

```
UPDATE Staff
SET MonthlySalary = MonthlySalary * 1.1;
```

(3 marks)

f) Staff named Ricky has changed address from Johor Bharu to Sri Petaling, make these changes

```
UPDATE Staff
SET Address = 'Sri Petaling'
WHERE Name = 'Ricky' AND Address = 'Johor Bharu';
```

(3 marks)

g) Staff named Eugene had resigned from the company, delete his record.

```
DELETE FROM Staff
WHERE Name = 'Eugene';
```

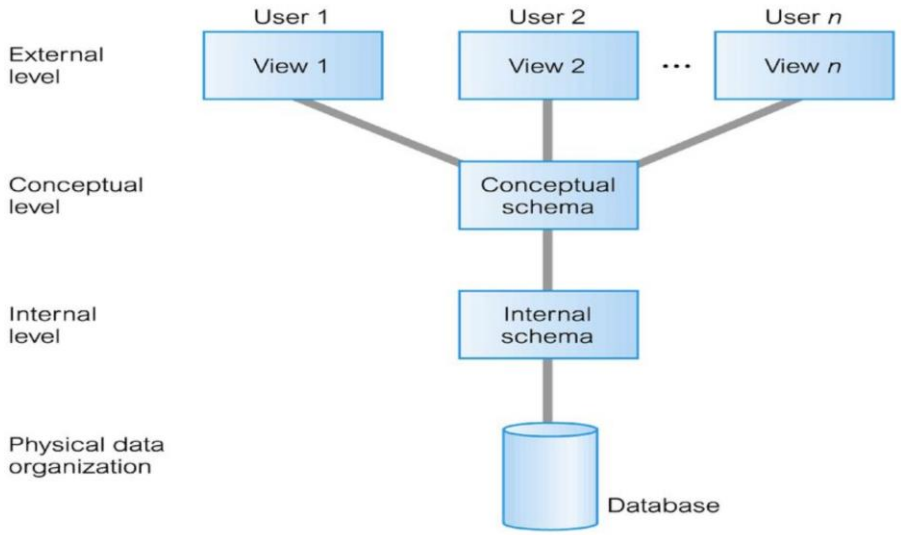
(3 marks)

h) Write SQL DDL to create Staff table above

```
CREATE TABLE Staff (
    Staff_ID INT PRIMARY KEY,
    Name NVARCHAR(255),
    Gender NVARCHAR(10),
    Address NVARCHAR(255),
    MonthlySalary DECIMAL(10, 2),
    Dept_ID NVARCHAR(50),
    CONSTRAINT fk_Dept FOREIGN KEY (Dept_ID) REFERENCES Department(Dept_ID)
);
```

(4 marks)
(Total: 25 marks)

2) Using an appropriate diagram, briefly explain the three levels of data abstraction in the ANSI-SPARC database architecture.



At the external level, users access the database through customized views, which present a tailored subset of data to meet specific user or group needs. Views simplify user interactions by hiding unnecessary details of the database structure, allowing users to focus on their specific requirements.

The conceptual level defines the logical structure of the entire database, outlining what data is stored, how it's related, and any integrity constraints. It provides a high-level abstraction, independent of physical storage details. Database administrators and designers work at this level to create a schema used by all users, focusing on the overall organization and relationships within the data.

The internal level handles the physical storage and access methods for data, addressing aspects like storage structures and formats. It emphasizes optimizing performance and storage efficiency. The database management system (DBMS) translates the logical schema to the physical schema for storage and retrieval. Users and application developers usually don't interact directly with the internal level; the DBMS takes care of managing these details.