DATABASE

COMS W1001
Introduction to Information Science

Boyi Xie

Today's Topics

- Database System Applications
- Database Systems versus File Systems
- View of Data
- Database Architecture
- E-R Model
- Relational Model
- Fundamental Operations
- A Sample Relational Database

Database System Applications

- Banking
  - customer information, accounts, transactions
- Airlines
  - reservation, schedule
- Universities
  - student information, course registration
- Telecommunication
  - records of calls made, monthly bills
- Finance
  - purchases of financial instruments such as stocks
- Sales
  - product, customer, purchase information
- Manufacturing
  - inventories, orders, supply chain management
- Human resources
  - information about employees, salaries, payroll

Database Systems vs File Systems

- Using Files for data management
- Data redundancy and inconsistency
- Difficulty in accessing data
- Data isolation

View of Data

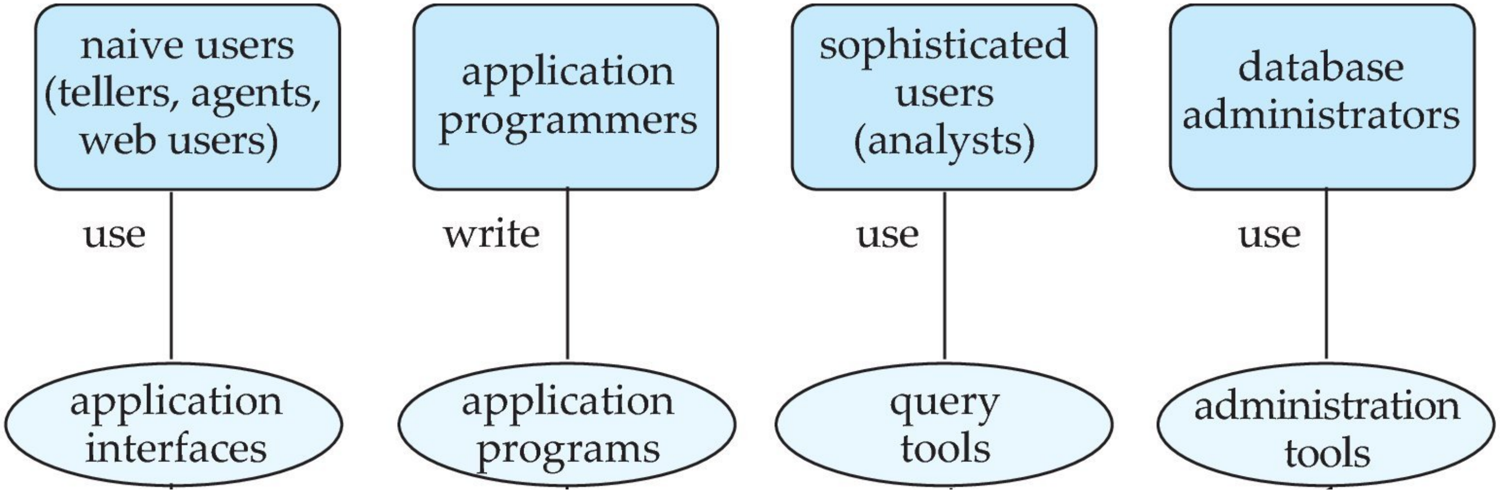
- Data Abstraction
  - Physical level – the lowest level of abstraction describes how the data are actually stored.
  - Logical level – the next-higher level of abstraction describes what data are stored in the database, and what relationships exist among those data.
  - View level – the highest level of abstraction describes only part of the entire database.
- Instances and Schemas
  - Instance – the collection of information stored in the database at a particular moment is called an instance of the database.
  - Schema – the overall design of the database is called the database schema. Schemas are changed infrequently.

Database Languages

- Data-Definition Language
  - Specify the database schema
  - Table definition and creation
  - Metadata – data about data
  - E.g. CREATE TABLE
- Data-Manipulation Language
  - Retrieval of information stored in the database
  - Insertion of new information into database
  - Deletion of information from the database
  - Modification of information stored in the database
  - E.g. SELECT

Database Users and Administrators

- People work with a database can be categorized as database users or database administrators



Transaction Management

- Often, several operations on the database form a single logical unit of work.
  - E.g. a funds transfer, one account is debited and another account is credited. Either both the credit and debit occur, or that neither occur.

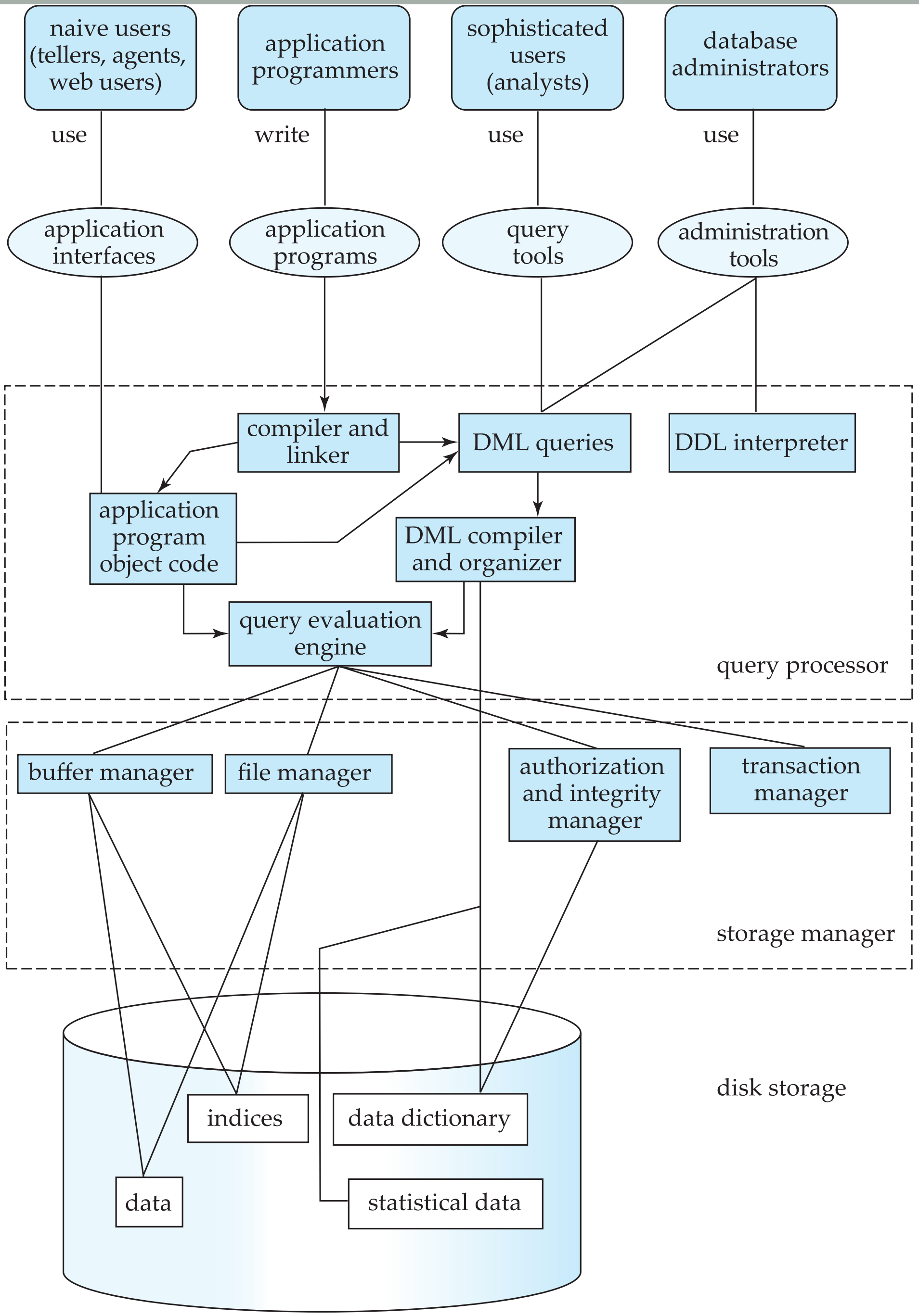
- A transaction is a **unit** of program execution that accesses and possibly updates various data items
  - Atomicity – either all operations of the transaction are reflected properly in the database, or none are
  - Consistency – the correctness of the transaction and the preservation of the consistency of the database
  - Isolation – even though multiple transactions may execute concurrently, each transaction is unaware of other transactions
  - Durability – after a transaction completes successfully, the changes it has made to the database persist, even if there are system failures

Database System Structure

- Storage Manager
  - Authorization and integrity manager
    - Tests for the satisfaction of integrity constraints and checks the authority of users to access data
  - Transaction manager
    - Ensures that the database remains in a consistent (correct) state despite system failures, and the concurrent transaction executions proceed without conflicting
  - File manager
    - Manages the allocation of space on disk storage and the data structures used to represent information stored on disk
  - Buffer manager
    - Responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory

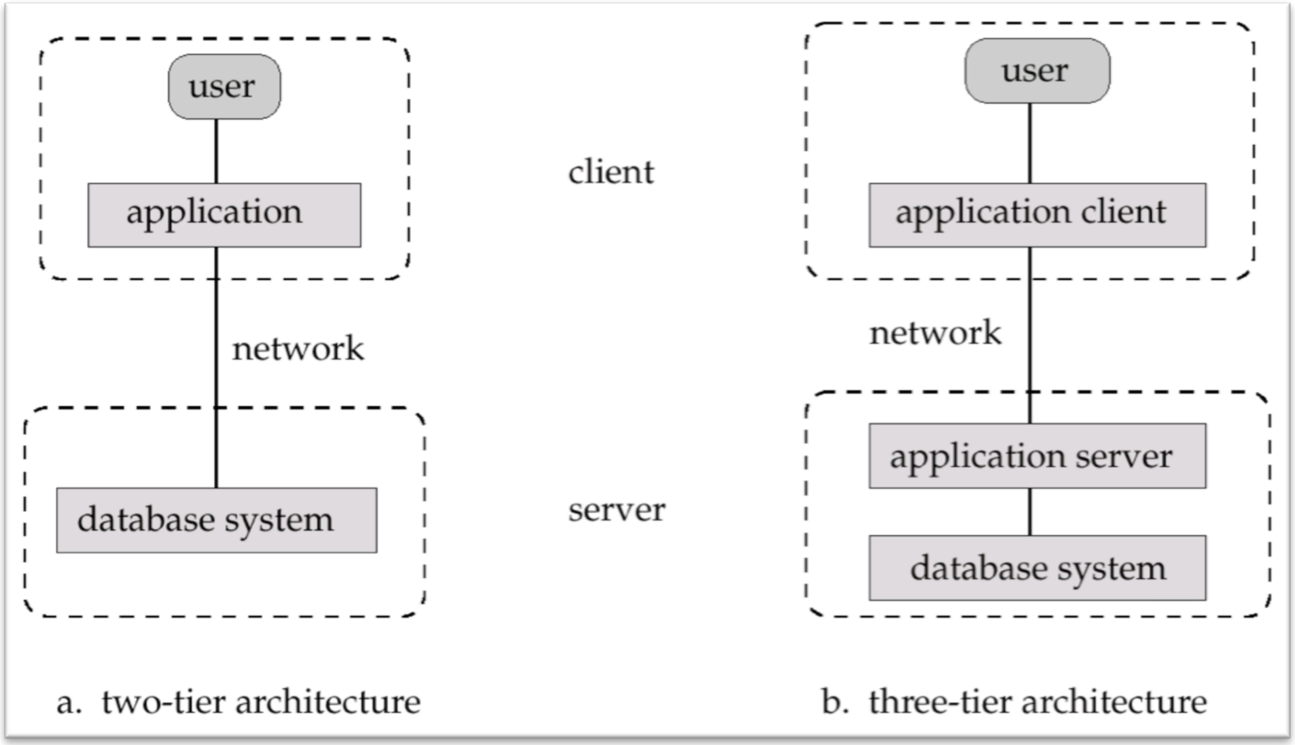
Database System Structure

- The Query Processor
  - DDL interpreter
    - Interpret DDL statements and records the definitions in the data dictionary
  - DML compiler
    - Translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands
  - Query evaluation engine
    - Executes low-level instructions generated by the DML compiler


naive users (tellers, agents, web users)
application programmers
sophisticated users (analysts)
database administrators
use
write
use
use
application interfaces
application programs
query tools
administration tools
compiler and linker
DML queries
DDL interpreter
application program object code
DML compiler and organizer
query evaluation engine
query processor
buffer manager
file manager
authorization and integrity manager
transaction manager
storage manager
disk storage
indices
data dictionary
data
statistical data

Application Architectures

- Two-tier architecture
- Three-tier architecture



Data Model

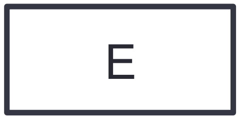
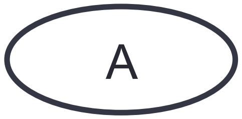
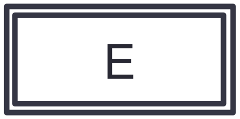
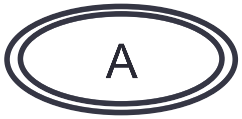
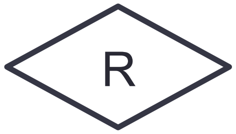
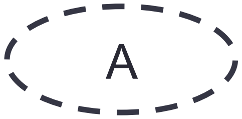
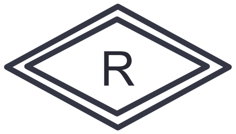
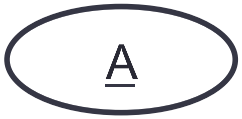
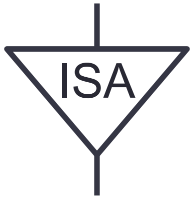
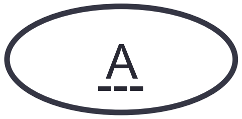
- Entity-Relationship (E-R) model
  - The E-R model perceives the real world as consisting of basic objects, called entities, and relationships among these objects.
  - It is very useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema.
  - Many database design tools draw on concepts from the E-R model.
- Relational Model
  - The relational model is today the primary data model for commercial data-processing applications.
  - A relational database consists of a collection of tables
  - A row in a table represents a relationship among a set of values

Database designers often formulate database schema design by first modeling data at a high level, using the E-R model, and then translating it into the relational model.

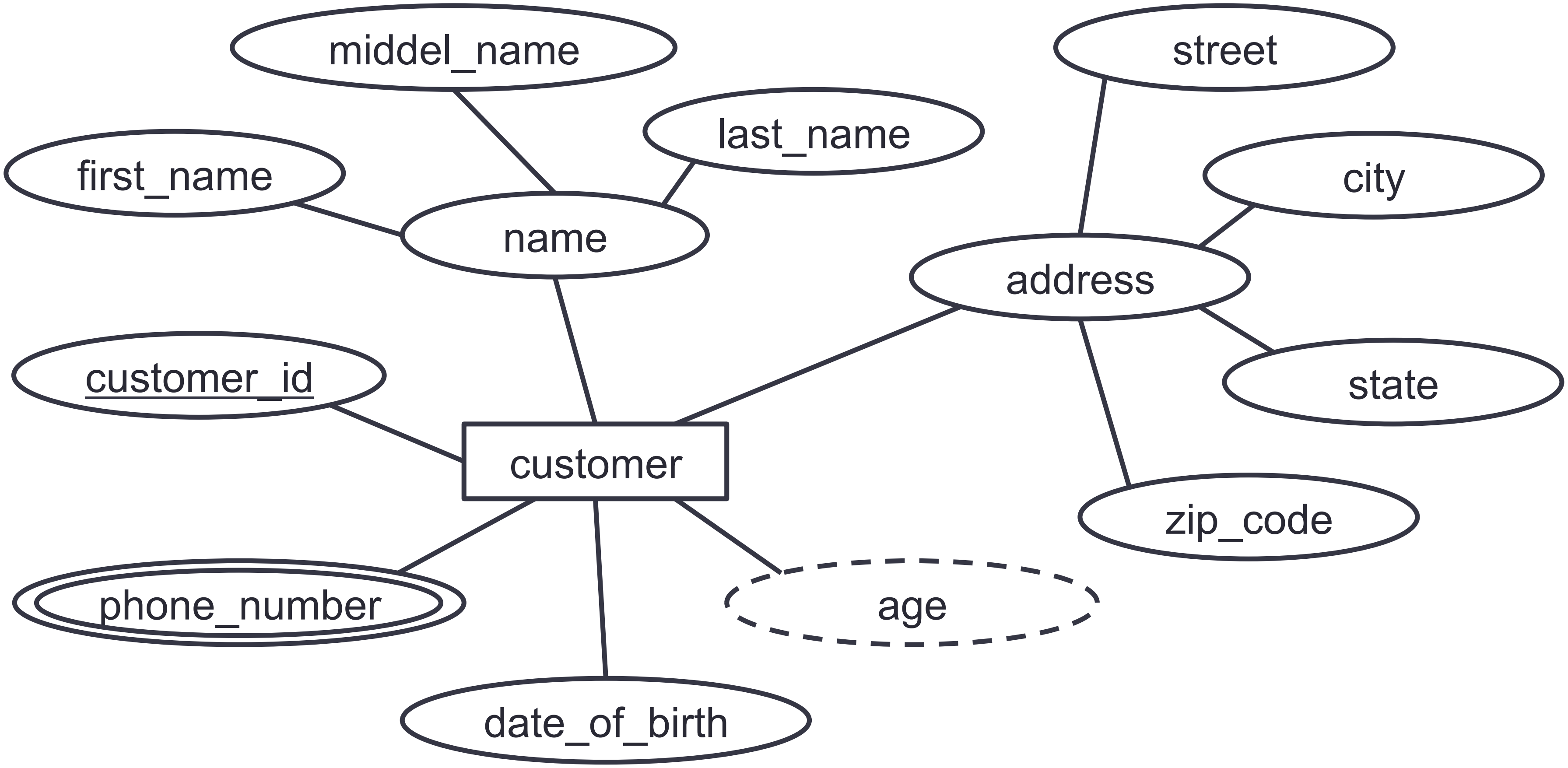
Entity-Relationship Model

- The E-R model employs three basic notions
  - Entity sets
    - An entity is a “thing” or “object” in the real world that is distinguishable from all other objects
    - An entity set is a set of entities of the same type that share the same properties, or attributes
  - Relationship sets
    - A relationship is an association among several entities
    - The function that an entity plays in a relationship is called that entity’s role
    - A relationship may also have attributes called descriptive attributes
  - Attributes
    - Simple and composite attributes
      - E.g. person name is a composite attribute consisting of first-name, middle-name, and last-name
    - Single-valued and multivalued attributes
      - E.g. phone number can be a multivalued attribute because a person may have zero, one, or several phone numbers
    - Derived attributes
      - E.g. age is a derived attribute, which can be computed from date-of-birth and the current date. In this case, date-of-birth may be referred to as a base attribute

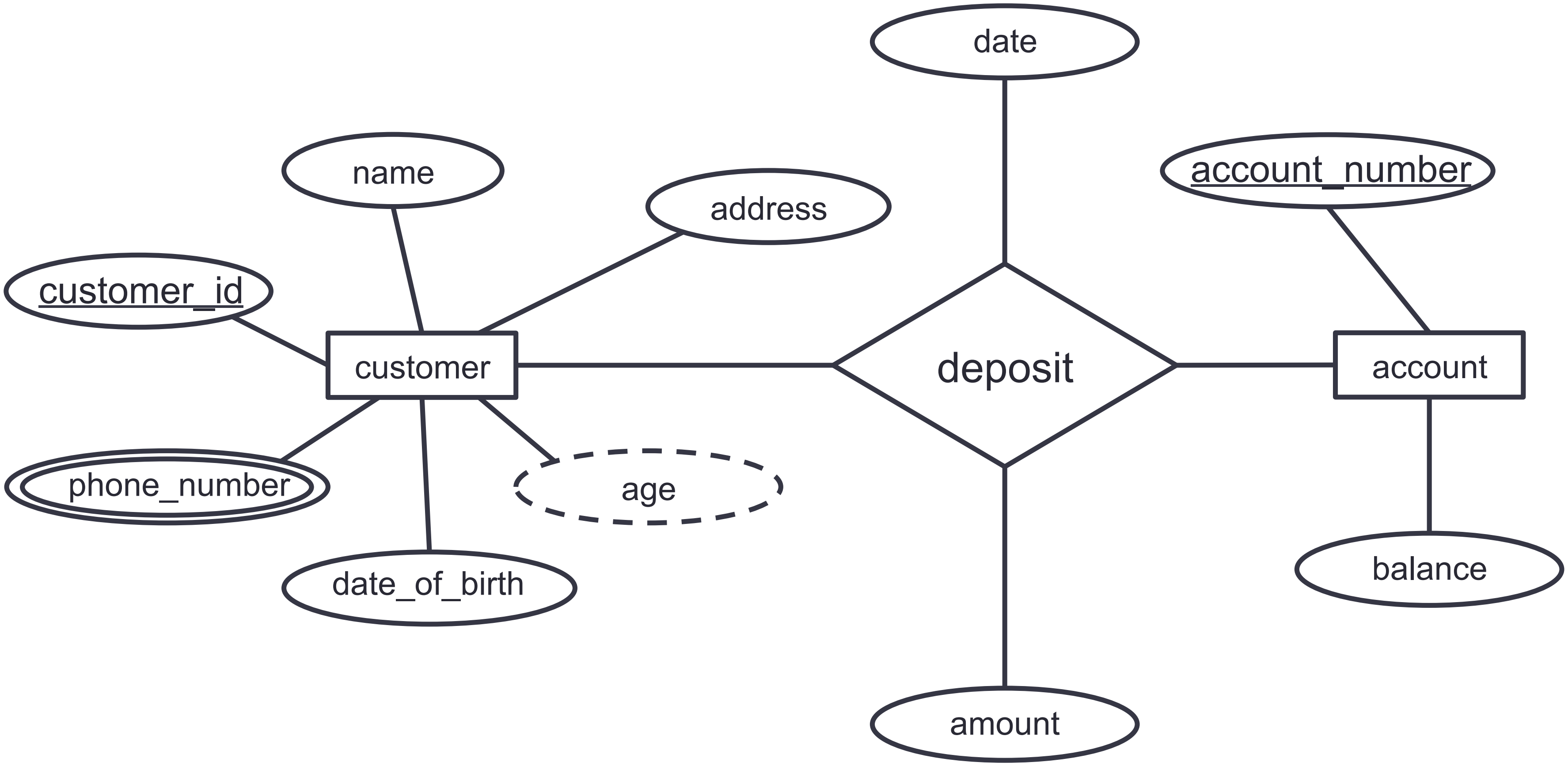
Symbols used in the E-R notation

	entity set		attribute
	weak entity set		multivalued attribute
	relationship set		derived attribute
	relationship set for weak entity set		primary key
	Is-A (specialization or generalization)		discriminating attribute of weak entity set

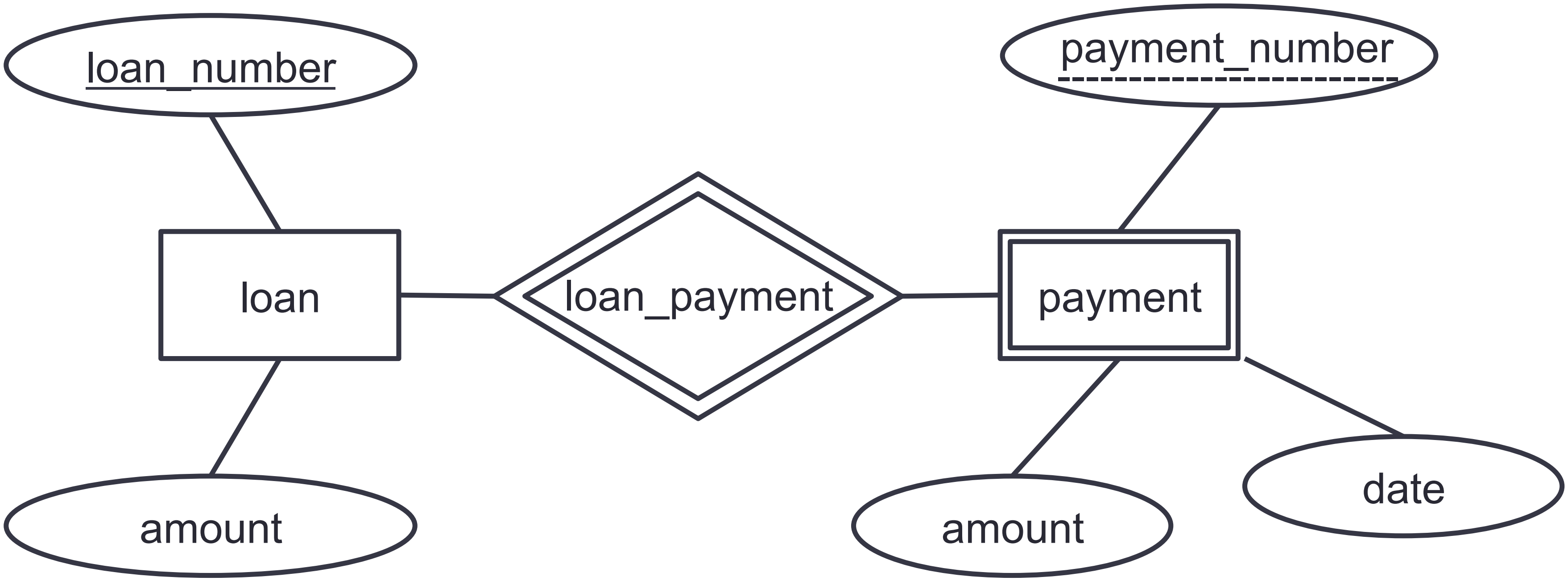
E-R diagram with composite, multivalued, and derived attributes



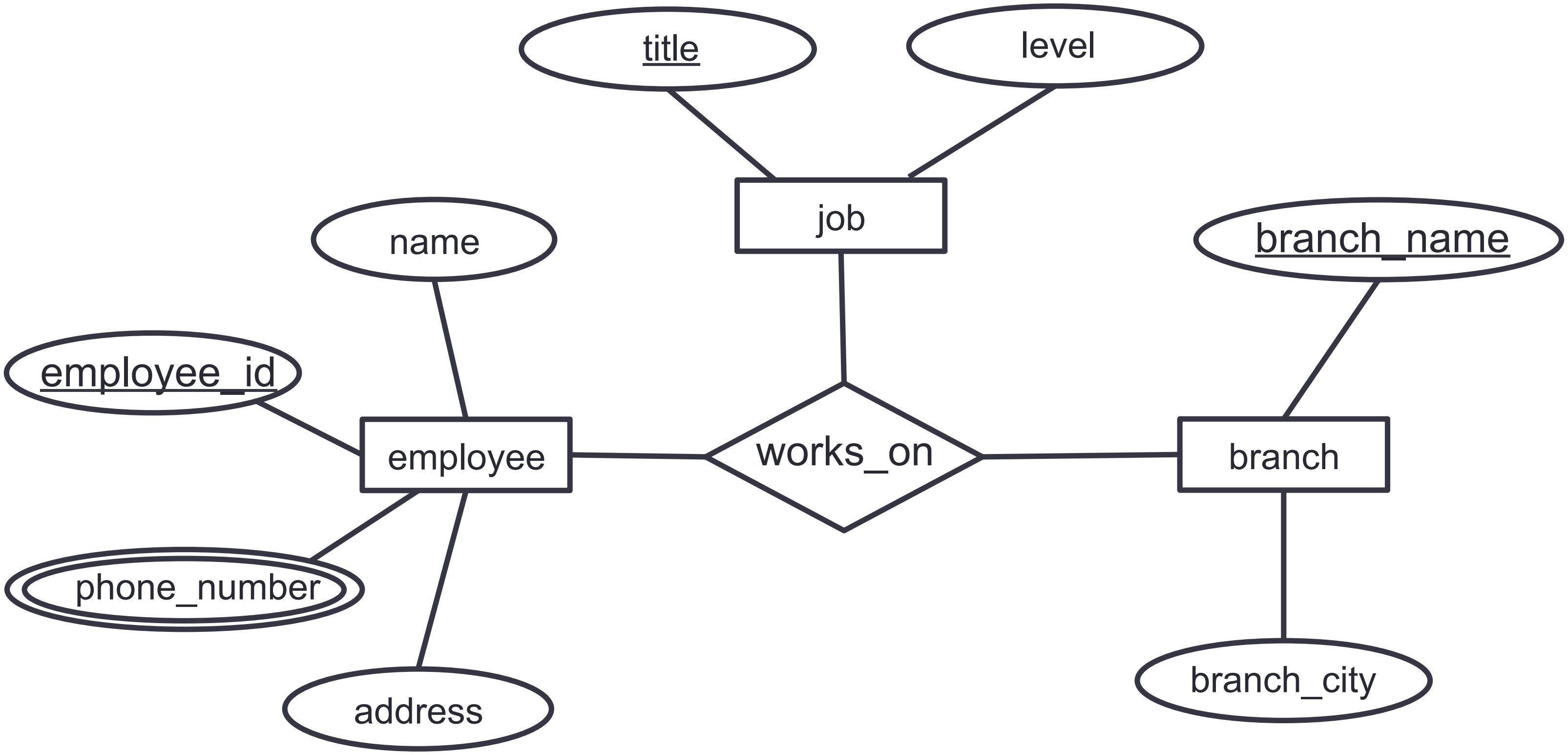
E-R diagram with attributes attached to a relationship set



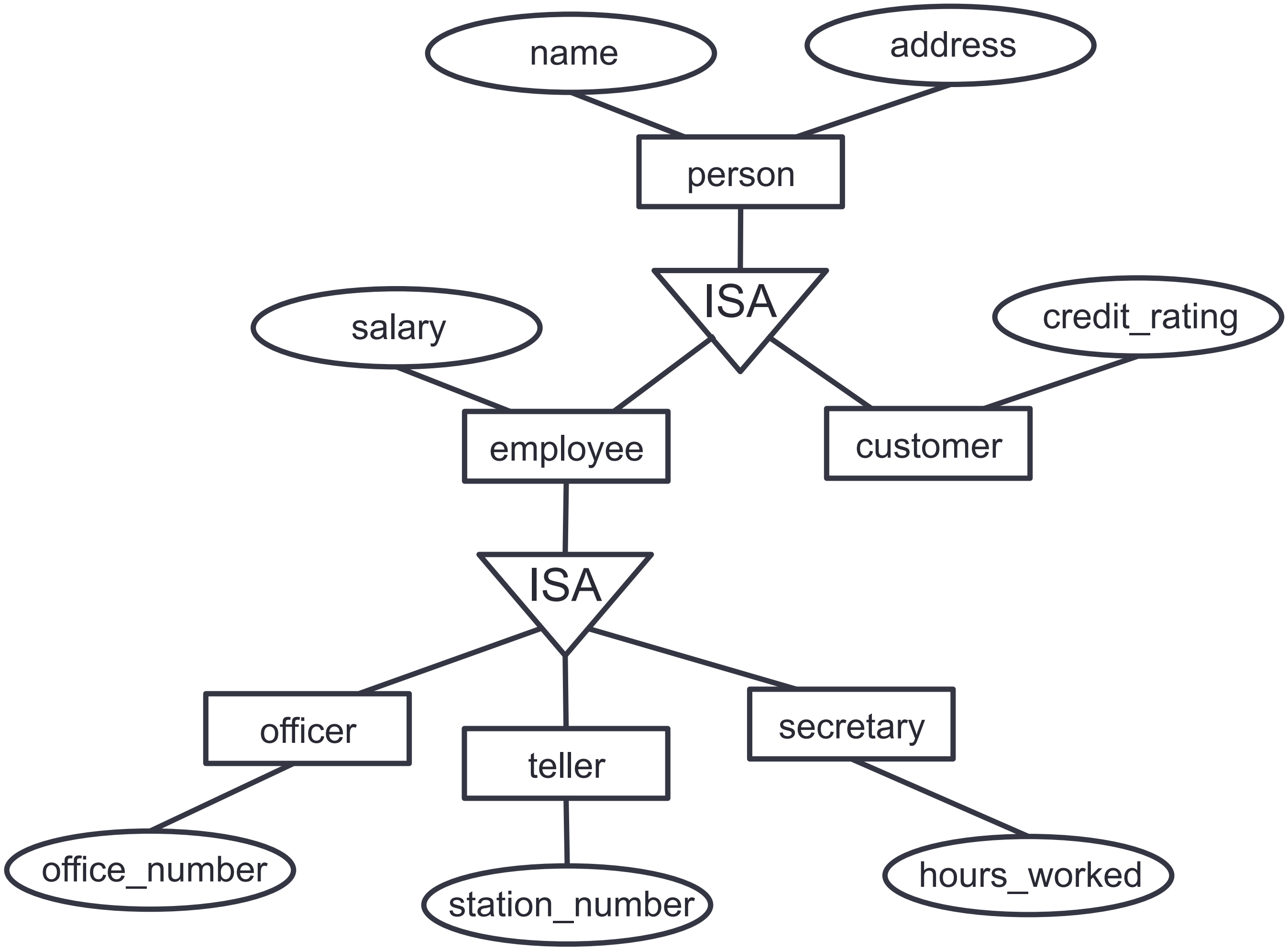
E-R diagram with a weak entity set



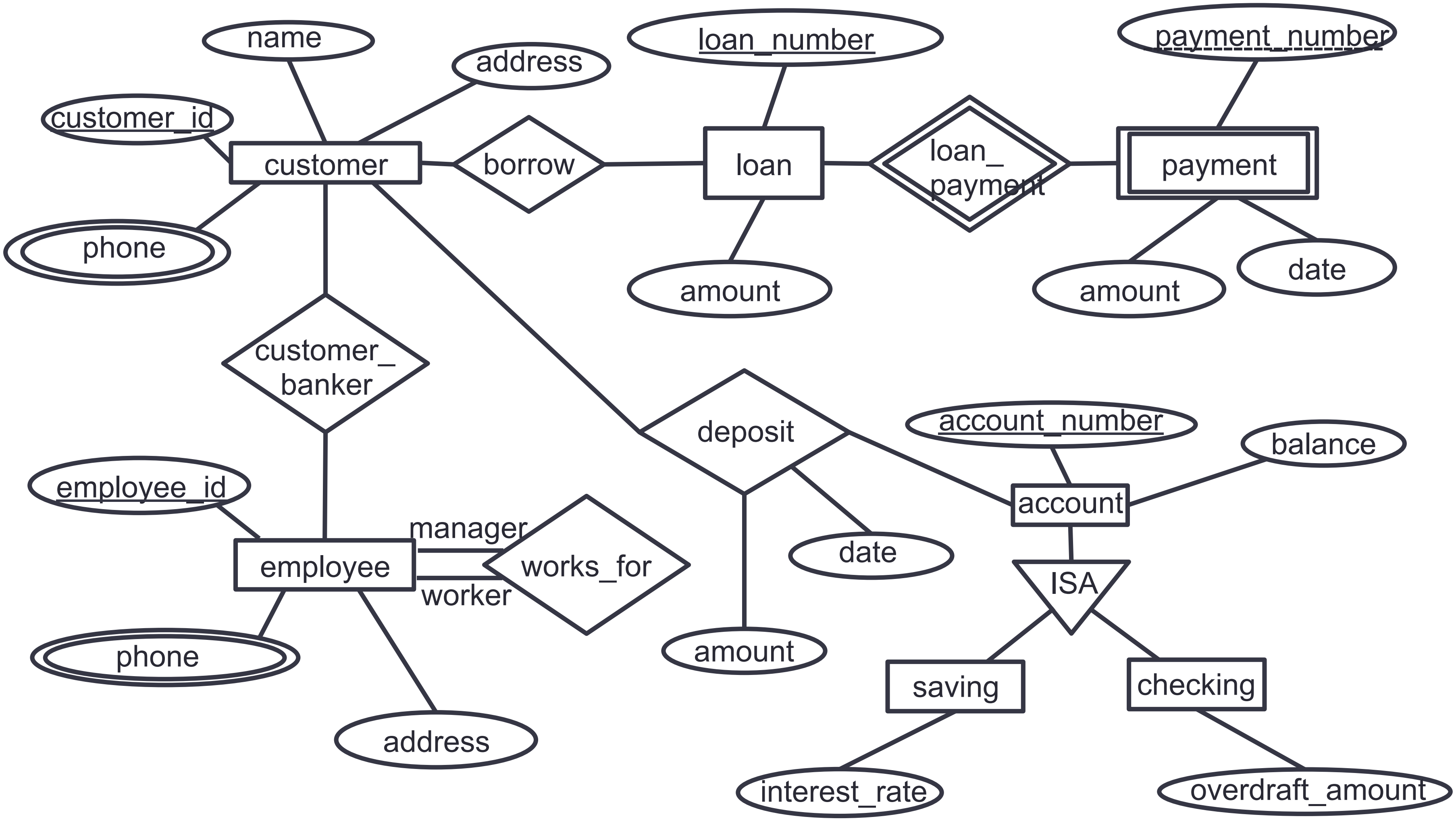
E-R diagram with a ternary relationship



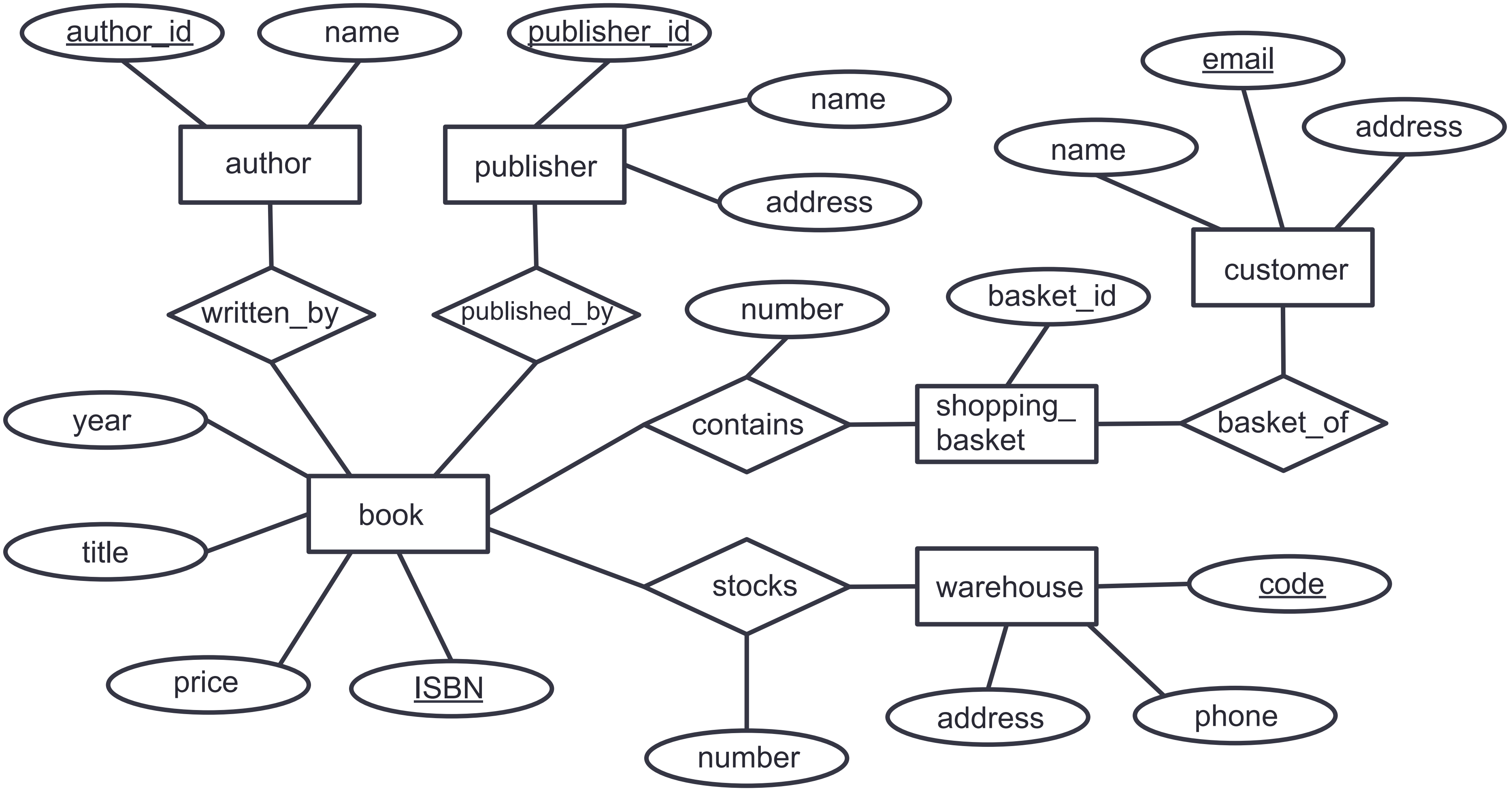
E-R diagram with specialization and generalization



E-R diagram for a banking enterprise



E-R diagram for an e-commerce enterprise



Relational Model

- The relational mode is today the primary data model for commercial data-processing applications
- A relational database consists of a collection of tables, each of which is assigned a unique name
- The headers (or columns) of a table are attributes
- A row in a table is a tuple of the attributes, and represents a record
- We can represent Entity-Relationship diagrams by tables in relational model

Fundamental Operations

- The Project Operation (pi π)
  - $\pi_{\text{loan-number,amount}}(\text{loan})$

- The Select Operation (sigma σ)
  - $\sigma_{\text{amount}>1200}(\text{loan})$
  - $\sigma_{\text{branch-name=“Perryridge”}}(\text{loan})$
  - $\sigma_{\text{branch-name=“Perryridge”} \wedge \text{amount}>1200}(\text{loan})$

- Composition of Relational Operation
  - $\pi_{\text{customer-name}}(\sigma_{\text{customer-city}}\text{=“Harrison”}(\text{customer}))$

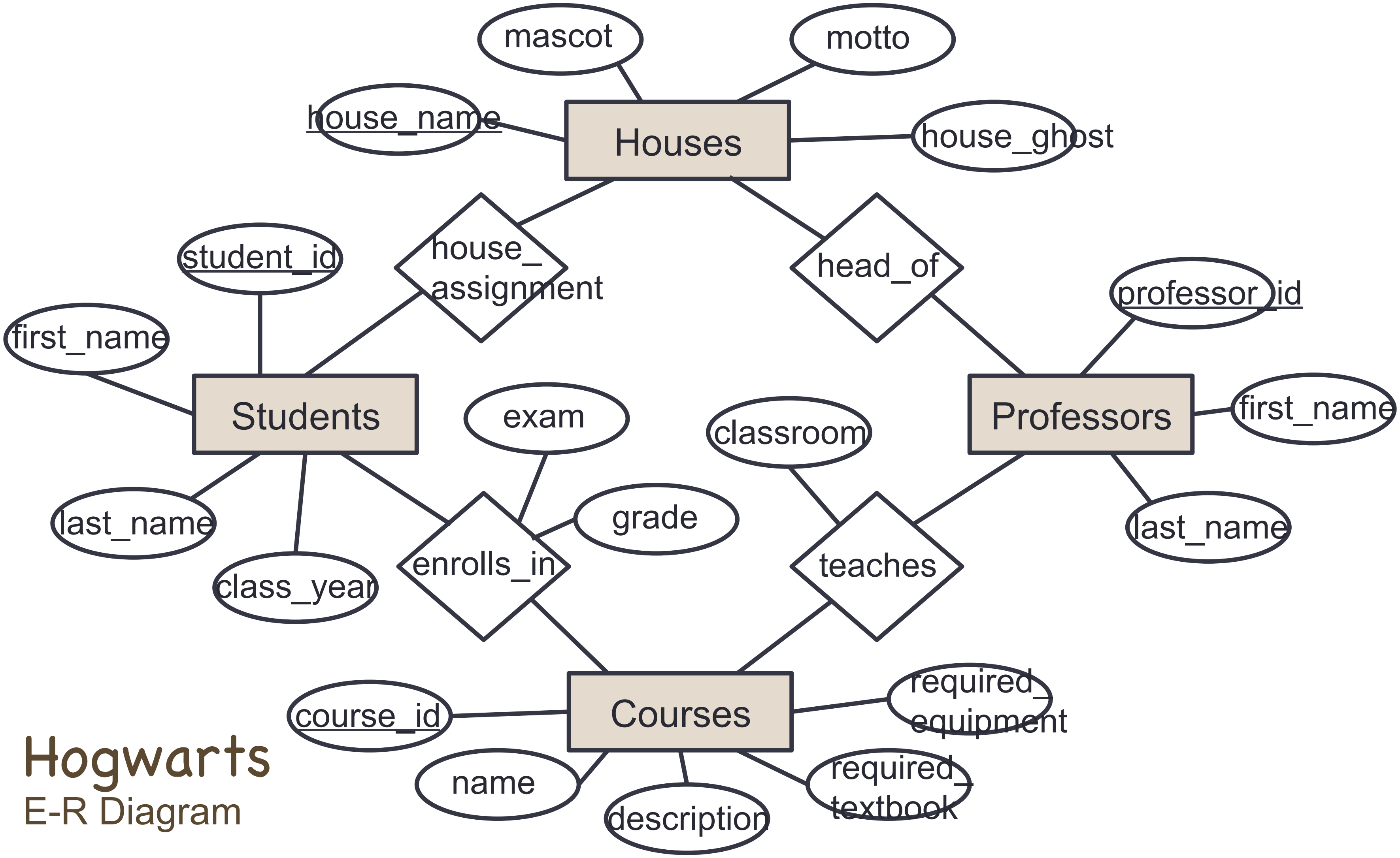
- The Rename Operation (rho ρ)
  - $\rho_{x(A1,A2,\ldots,An)}(E)$
  - returns the result of expression E under the name x, with the attributes renamed to A1,A2,…,An.

- Join Operation (⋈)
  - Combine rows from two or more tables, based on a common field between them.

A Sample Relational Database

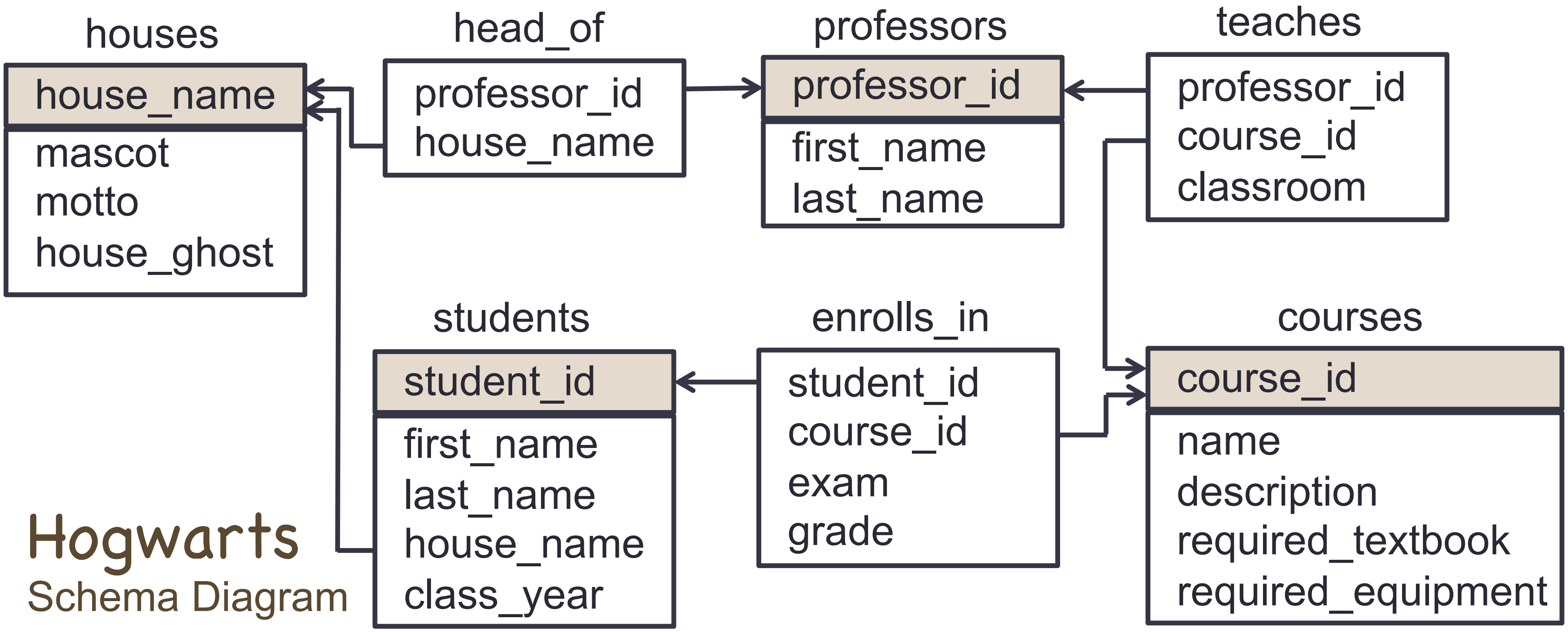
- This sample database will address the following topics
  - E-R diagram to schema diagrams for relational database
  - Create and delete a database (schema)
  - Create and delete a table
  - Data types
  - Insert, update and delete records (rows in a table)
  - Queries
    - Project operation
    - Select operation
    - Rename operation
    - Join operation
    - Others

E-R Diagram



Hogwarts
E-R Diagram

Schema Diagram



Hogwarts
Schema Diagram

Creation and Deletion of a Database

- Create a database (schema)

```
CREATE DATABASE hogwarts;
```

- Delete a database (schema)

```
DROP DATABASE hogwarts;
```

- Show Databases

```
SHOW DATABASES;
```

- Show Tables

```
SHOW TABLES;
```

Create and Delete a Table, Data Types

- Create a table

```
CREATE TABLE students (
        student_id INT NOT NULL
AUTO_INCREMENT,
        first_name VARCHAR(255),
        last_name VARCHAR(255),
        house_name VARCHAR(100),
        class_year INT,
        PRIMARY KEY (student_id)
);
```

- Delete all records in a table

```
TRUNCATE TABLE students;
```

- Delete a table

```
DROP TABLE students;
```

More about data types: http://dev.mysql.com/doc/refman/5.6/en/data-types.html

Insert, Update, and Delete a Record

- Insert a record

```
INSERT INTO enrolls_in
(student_id, course_id, exam, grade)
VALUES ("1", "1", "86", "B");
```

- Update a record

```
UPDATE enrolls_in SET grade = 'A'
WHERE student_id = '1' AND course_id = '5';
```

- Delete a record

```
DELETE FROM enrolls_in
WHERE student_id = '9' AND course_id = '5';
```

Basic Queries

- Select all records in a table

```
SELECT * FROM students;
```

- Project partial attributes

```
SELECT first_name, last_name, house_name
FROM students;
```

- Create View

```
CREATE VIEW house_assignments AS
SELECT first_name, last_name, house_name
FROM students;
```

Rename Tables and Attributes

- Rename tables

```
SELECT * FROM head_of AS h, professors AS p
WHERE h.professor_id = p.professor_id;
```

- Rename attributes

```
SELECT p.first_name AS "First Name", p.last_name
AS "Last Name", h.house_name AS "House"
FROM head_of AS h, professors AS p
WHERE h.professor_id = p.professor_id;
```

Join, Natural Join, Outer Join

- Join

```
SELECT * FROM head_of JOIN professors;
```

- Join based on conditions

```
SELECT * FROM head_of JOIN professors
ON head_of.professor_id = professors.professor_id;
```

- Natural join

```
SELECT * FROM head_of NATURAL JOIN professors;
```

- Outer join
  - Left join

```
SELECT * FROM professors LEFT JOIN head_of
ON head_of.professor_id = professors.professor_id;
```

  - Right join

```
SELECT * FROM head_of RIGHT JOIN professors
ON head_of.professor_id = professors.professor_id;
```

Group By, Order By

- Group By

```
SELECT e.student_id, s.first_name, s.last_name, COUNT(*)
FROM enrolls_in AS e, students AS s
WHERE e.student_id = s.student_id
GROUP BY e.student_id;
```

- Order By

```
SELECT s.first_name, s.last_name, AVG(e.exam)
FROM enrolls_in AS e, students AS s, courses AS c
WHERE e.course_id = c.course_id AND e.student_id = s.student_id
GROUP BY s.first_name, s.last_name
ORDER BY AVG(e.exam) DESC;
```

Other Useful Commands

- DISTINCT

```
SELECT DISTINCT e.student_id, s.first_name, s.last_name
FROM enrolls_in AS e, students AS s
WHERE e.student_id = s.student_id;
```

- COUNT

```
SELECT e.student_id, s.first_name, s.last_name, COUNT(*)
FROM enrolls_in AS e, students AS s
WHERE e.student_id = s.student_id
GROUP BY e.student_id;
```

- AVG

```
SELECT s.first_name, s.last_name, AVG(e.exam)
FROM enrolls_in AS e, students AS s, courses AS c
WHERE e.course_id = c.course_id AND e.student_id = s.student_id
GROUP BY s.first_name, s.last_name
ORDER BY AVG(e.exam) DESC;
```

References & Photo Credits

- Abraham Silberschatz, Henry F. Korth, S. Sudarshan. Database System Concepts. McGraw-Hill.
- Brookshear, J. Glenn (2011-04-13). Computer Science: An Overview (11th Edition). Prentice Hall.
- Harry Potter Wiki. http://harrypotter.wikia.com