Chapter 1: Introduction

Database System Concepts, 5th Ed.

Chapter 1: Introduction

- Purpose of Database Systems

- View of Data

- Database Languages

- Relational Databases

- Database Design

- Object-based and semistructured databases

- Data Storage and Querying

- Transaction Management

- Database Architecture

- Database Users and Administrators

- Overall Structure

- History of Database Systems

Database Management System (DBMS)

- DBMS contains information about a particular enterprise
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use

- Database Applications:
  - Banking: all transactions
  - Airlines: reservations, schedules
  - Universities: registration, grades
  - Sales: customers, products, purchases
  - Online retailers: order tracking, customized recommendations
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions

- Databases touch all aspects of our lives

Purpose of Database Systems

- In the early days, database applications were built directly on top of file systems

- Drawbacks of using file systems to store data:
  - Data redundancy and inconsistency
    - Multiple file formats, duplication of information in different files
  - Difficulty in accessing data
    - Need to write a new program to carry out each new task
  - Data isolation — multiple files and formats
  - Integrity problems
    - Integrity constraints (e.g. account balance > 0) become "buried" in program code rather than being stated explicitly
    - Hard to add new constraints or change existing ones

Purpose of Database Systems (Cont.)

- Drawbacks of using file systems (cont.)

  - Atomicity of updates

    - Failures may leave database in an inconsistent state with partial updates carried out

    - Example: Transfer of funds from one account to another should either complete or not happen at all

  - Concurrent access by multiple users

    - Concurrent accessed needed for performance

    - Uncontrolled concurrent accesses can lead to inconsistencies

      - Example: Two people reading a balance and updating it at the same time

  - Security problems

    - Hard to provide user access to some, but not all, data

- Database systems offer solutions to all the above problems

Levels of Abstraction

- **Physical level:** describes how a record (e.g., customer) is stored.

- **Logical level:** describes data stored in database, and the relationships among the data.

  type *customer* = **record**

            customer_id : string;
            customer_name : string;
            customer_street : string;
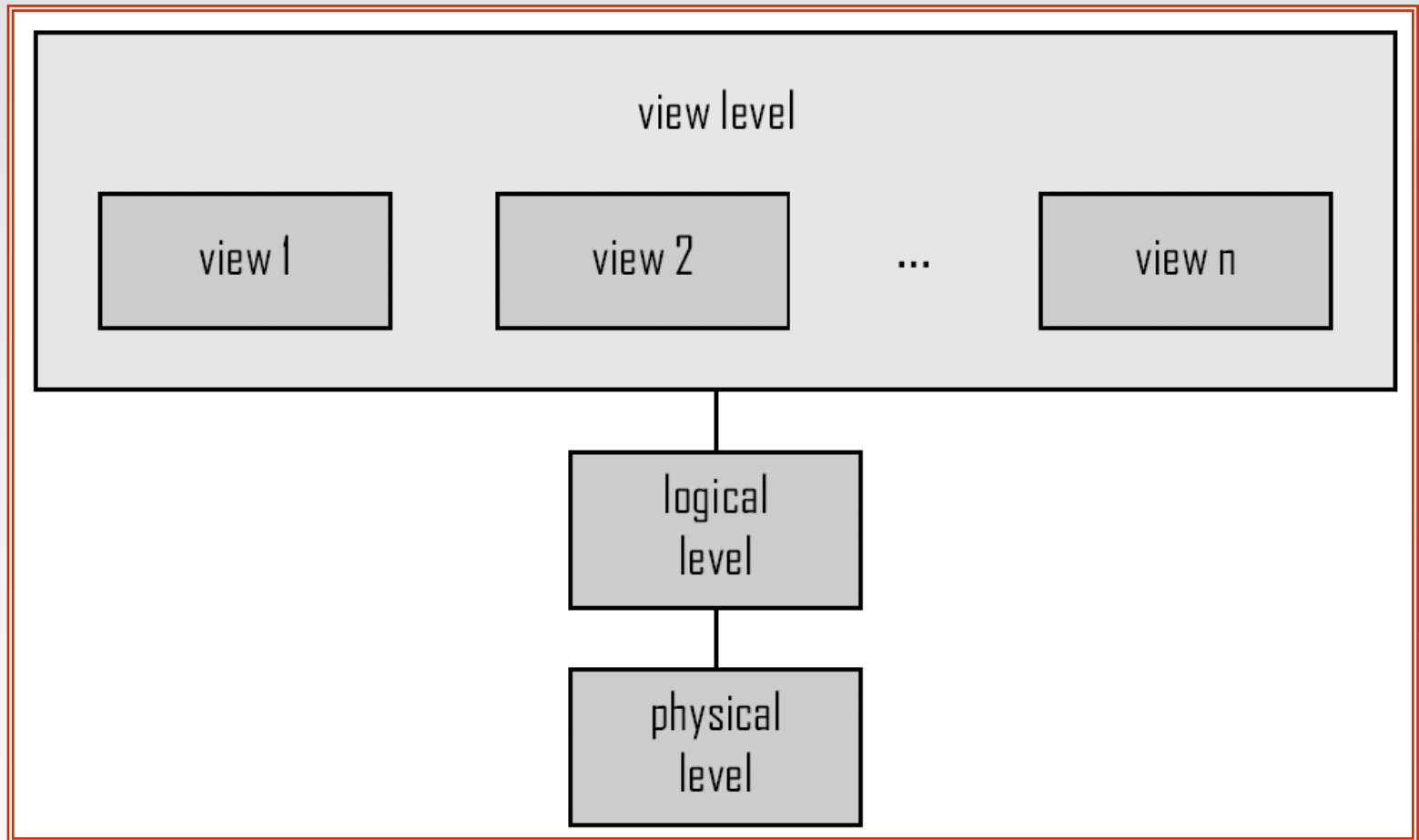   *customer_city* : integer;

      end

- **View level:** A way to hide: (a) details of data types and (b) information (such as an employee's salary) for security purposes.

View of Data

An architecture for a database system

Instances and Schemas

- Similar to types and variables in programming languages

- **Schema** – the logical structure of the database
  - Example: The database consists of information about a set of customers and accounts and the relationship between them)
  - Analogous to type information of a variable in a program
  - **Physical schema**: database design at the physical level
  - **Logical schema**: database design at the logical level

- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable

- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semistructured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
    - DML also known as **query language**

- Two classes of languages

    - **Procedural** – user specifies what data is required and how to get those data

    - **Declarative (nonprocedural)** – user specifies what data is required without specifying how to get those data

- SQL is the most widely used query language

Data Definition Language (DDL)

- Specification notation for defining the database schema

  Example: **create table** *account* (

                                              account-number   **char**(10),
                                              balance   **integer**)

- DDL compiler generates a set of tables stored in a *data dictionary*

- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Integrity constraints
    - Domain constraints
    - Referential integrity (**references** constraint in SQL)
    - Assertions
  - Authorization

- Data *storage and definition* language
  - Specifies the storage structure and access methods used

Relational Databases

- A relational database is based on the relational data model

- Data and relationships among the data is represented by a collection of tables

- Includes both a DML and a DDL

- Most commercial relational database systems employ the **SQL** query langue.

Relational Model

Attributes

- Example of tabular data in the relational model

customer_id	customer_name	customer_street	customer_city	account_number
192-83-7465	Johnson	12 Alma St.	Palo Alto	A-101
192-83-7465	Johnson	12 Alma St.	Palo Alto	A-201
677-89-9011	Hayes	3 Main St.	Harrison	A-102
182-73-6091	Turner	123 Putnam St.	Stamford	A-305
321-12-3123	Jones	100 Main St.	Harrison	A-217
336-66-9999	Lindsay	175 Park Ave.	Pittsfield	A-222
019-28-3746	Smith	72 North St.	Rye	A-201
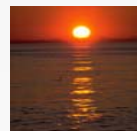
A Sample Relational Database

customer_id	customer_name	customer_street	customer_city
192-83-7465	Johnson	12 Alma St.	Palo Alto
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account_number	balance
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

customer_id	account_number
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table
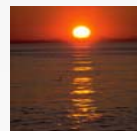
SQL

- **SQL**: widely used non-procedural language

  - Example: Find the name of the customer with customer-id 192-83-7465

    select *customer.customer_name*
    from *customer*
    where *customer.customer_id* = '192-83-7465'

  - Example: Find the balances of all accounts held by the customer with customer-id 192-83-7465

    select *account.balance*
    from *depositor*, *account*
    where *depositor.customer_id* = '192-83-7465' **and**
    depositor.account_number = *account.account_number*

- Application programs generally access databases through one of

  - Language extensions to allow embedded SQL

  - Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

Database Design

The process of designing the general structure of the database:

- Logical Design – Deciding on the database schema. Database design requires that we find a "good" collection of relation schemas.
    - Business decision – What attributes should we record in the database?
    - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?

- Physical Design – Deciding on the physical layout of the database

The Entity-Relationship Model

- Models an enterprise as a collection of *entities* and *relationships*
  - Entity: a "thing" or "object" in the enterprise that is distinguishable from other objects
    - Described by a set of *attributes*
  - Relationship: an association among several entities
- Represented diagrammatically by an *entity-relationship diagram:*
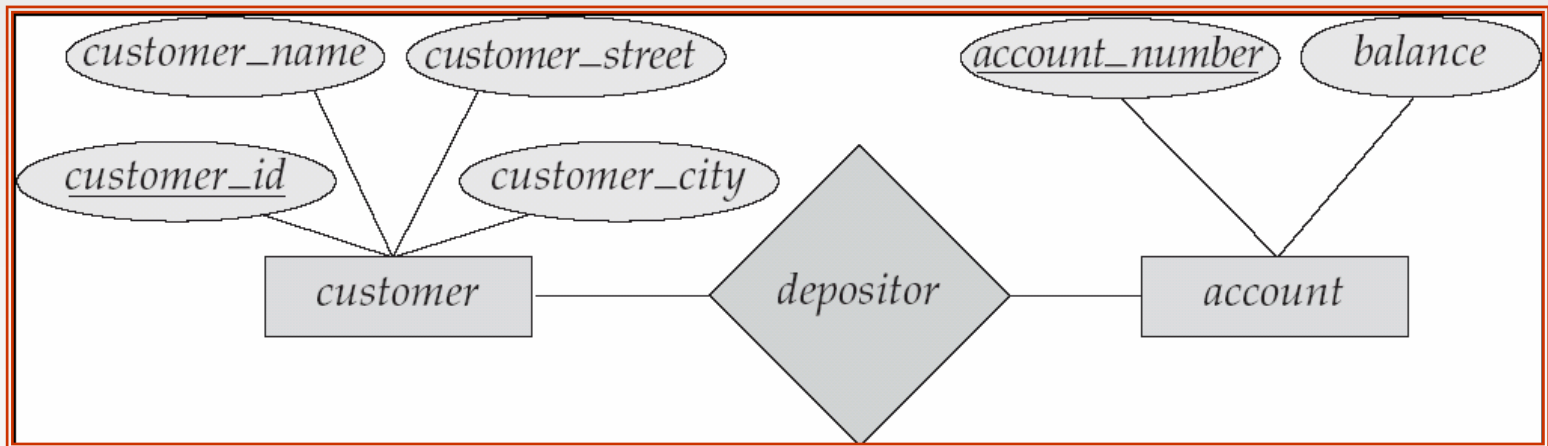
Object-Relational Data Models

- Extend the relational data model by including object orientation and constructs to deal with added data types.

- Allow attributes of tuples to have complex types, including non-atomic values such as nested relations.

- Preserve relational foundations, in particular the declarative access to data, while extending modeling power.

- Provide upward compatibility with existing relational languages.

XML: Extensible Markup Language

- Defined by the WWW Consortium (W3C)

- Originally intended as a document markup language not a database language

- The ability to specify new tags, and to create nested tag structures made XML a great way to exchange **data**, not just documents

- XML has become the basis for all new generation data interchange formats.

- A wide variety of tools is available for parsing, browsing and querying XML documents/data

Storage Management

- **Storage manager** is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.

- The storage manager is responsible to the following tasks:

  - Interaction with the file manager

  - Efficient storing, retrieving and updating of data

- Issues:

  - Storage access

  - File organization

  - Indexing and hashing

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation
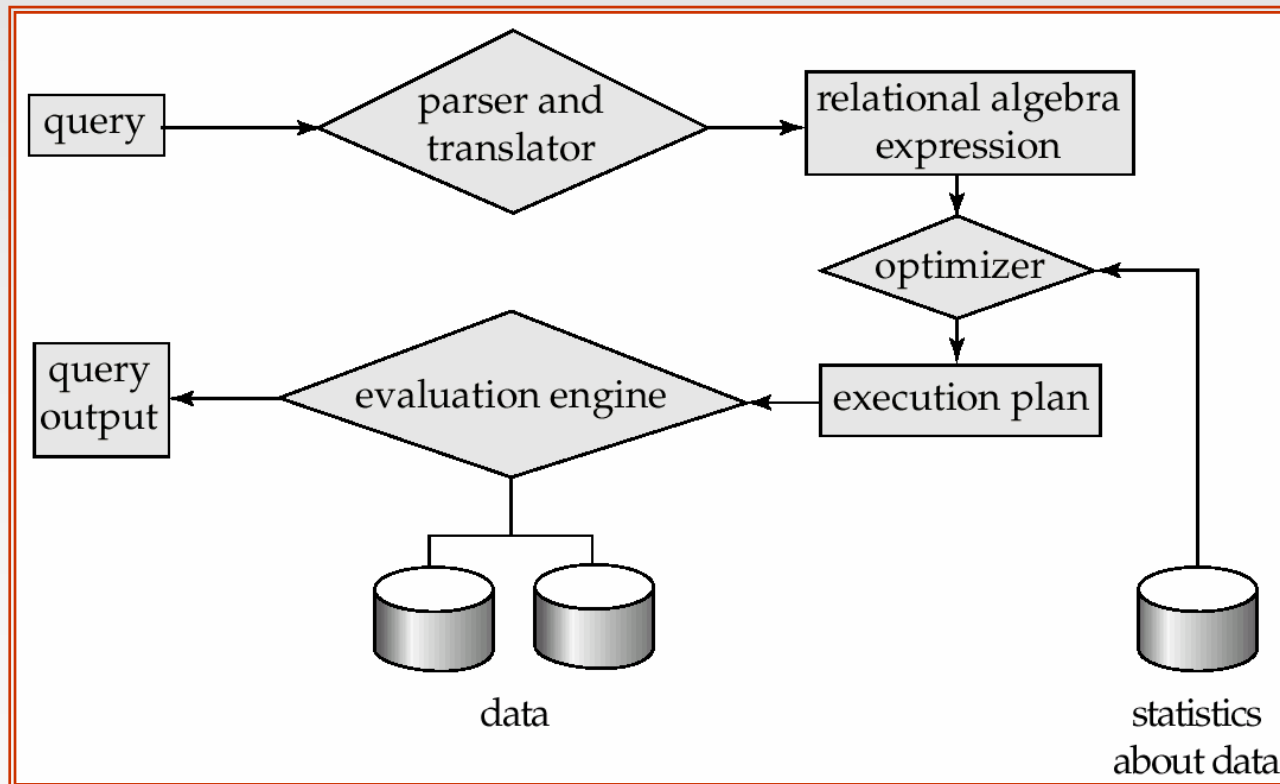
Query Processing (Cont.)

- Alternative ways of evaluating a given query

    - Equivalent expressions

    - Different algorithms for each operation

- Cost difference between a good and a bad way of evaluating a query can be enormous

- Need to estimate the cost of operations

    - Depends critically on statistical information about relations which the database must maintain

    - Need to estimate statistics for intermediate results to compute cost of complex expressions

Transaction Management

- A **transaction** is a collection of operations that performs a single logical function in a database application

- **Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.

- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.

Database Architecture

The architecture of a database systems is greatly influenced by

the underlying computer system on which the database is running:

- Centralized
- Client-server
- Parallel (multi-processor)
- Distributed

Database Users

Users are differentiated by the way they expect to interact with

the system

- **Application programmers** – interact with system through DML calls
- **Sophisticated users** – form requests in a database query language
- **Specialized users** – write specialized database applications that do not fit into the traditional data processing framework
- **Naïve users** – invoke one of the permanent application programs that have been written previously
  - Examples, people accessing database over the web, bank tellers, clerical staff

Database Administrator

- Coordinates all the activities of the database system; the database administrator has a good understanding of the enterprise's information resources and needs.

- Database administrator's duties include:

  - Schema definition
  - Storage structure and access method definition
  - Schema and physical organization modification
  - Granting user authority to access the database
  - Specifying integrity constraints
  - Acting as liaison with users
  - Monitoring performance and responding to changes in requirements

Overall System Structure
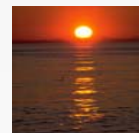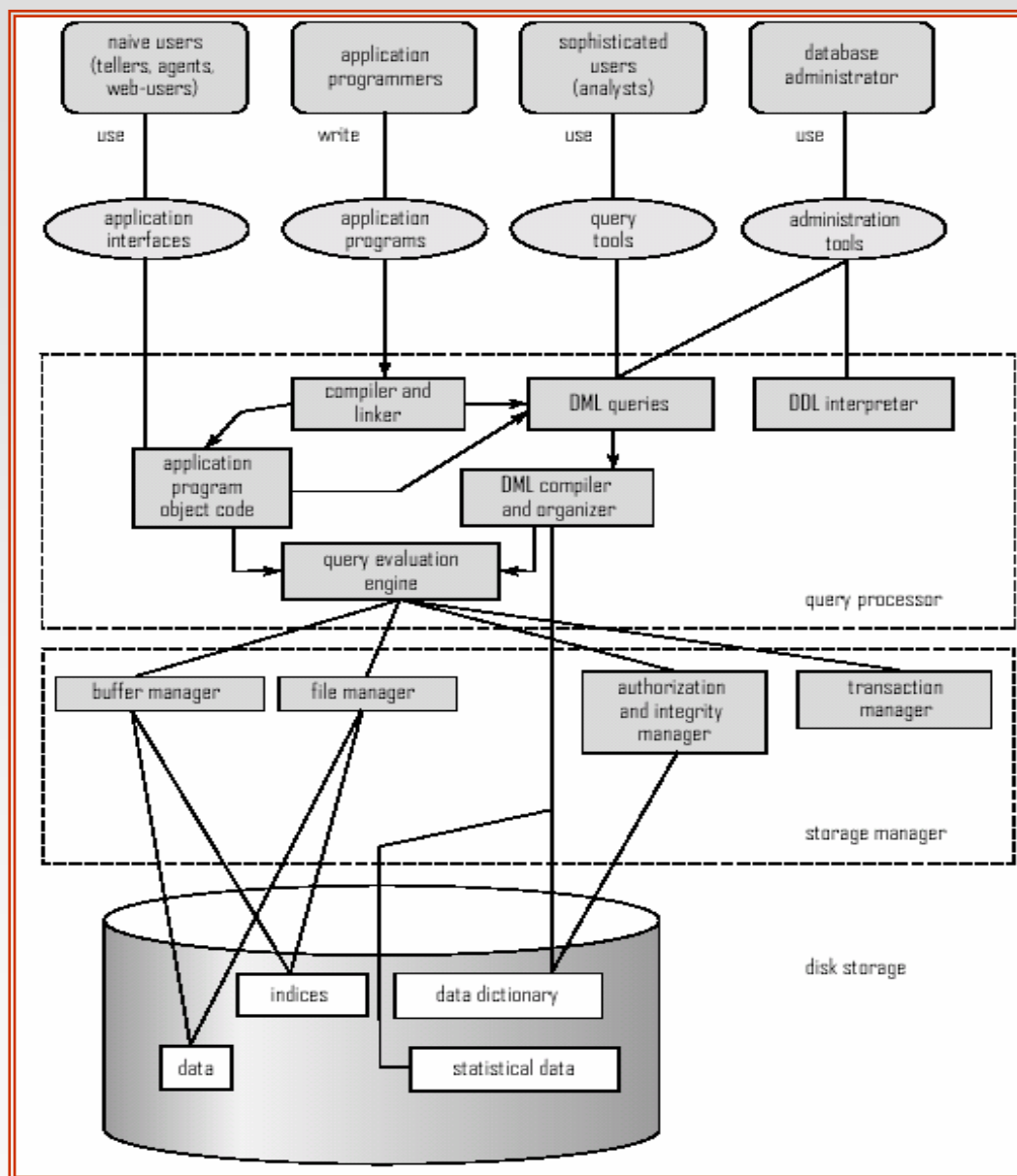
History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - Tapes provide only sequential access
  - Punched cards for input

- Late 1960s and 1970s:
  - Hard disks allow direct access to data
  - Network and hierarchical data models in widespread use
  - Ted Codd defines the relational data model
    - Would win the ACM Turing Award for this work
    - IBM Research begins System R prototype
    - UC Berkeley begins Ingres prototype
  - High-performance (for the era) transaction processing

History (cont.)

- 1980s:
  - Research relational prototypes evolve into commercial systems
    - SQL becomes industrial standard
  - Parallel and distributed database systems
  - Object-oriented database systems
- 1990s:
  - Large decision support and data-mining applications
  - Large multi-terabyte data warehouses
  - Emergence of Web commerce
- 2000s:
  - XML and XQuery standards
  - Automated database administration

End of Chapter 1

Database System Concepts, 5th Ed.

©Silberschatz, Korth and Sudarshan
See www.db-book.com for conditions on re-use

Figure 1.4

customer_id	account_number	balance
192-83-7465	A-101	500
192-83-7465	A-201	900
019-28-3746	A-215	700
677-89-9011	A-102	400
182-73-6091	A-305	350
321-12-3123	A-217	750
336-66-9999	A-222	700
019-28-3746	A-201	900

Figure 1.7

a. two-tier architecture
b. three-tier architecture