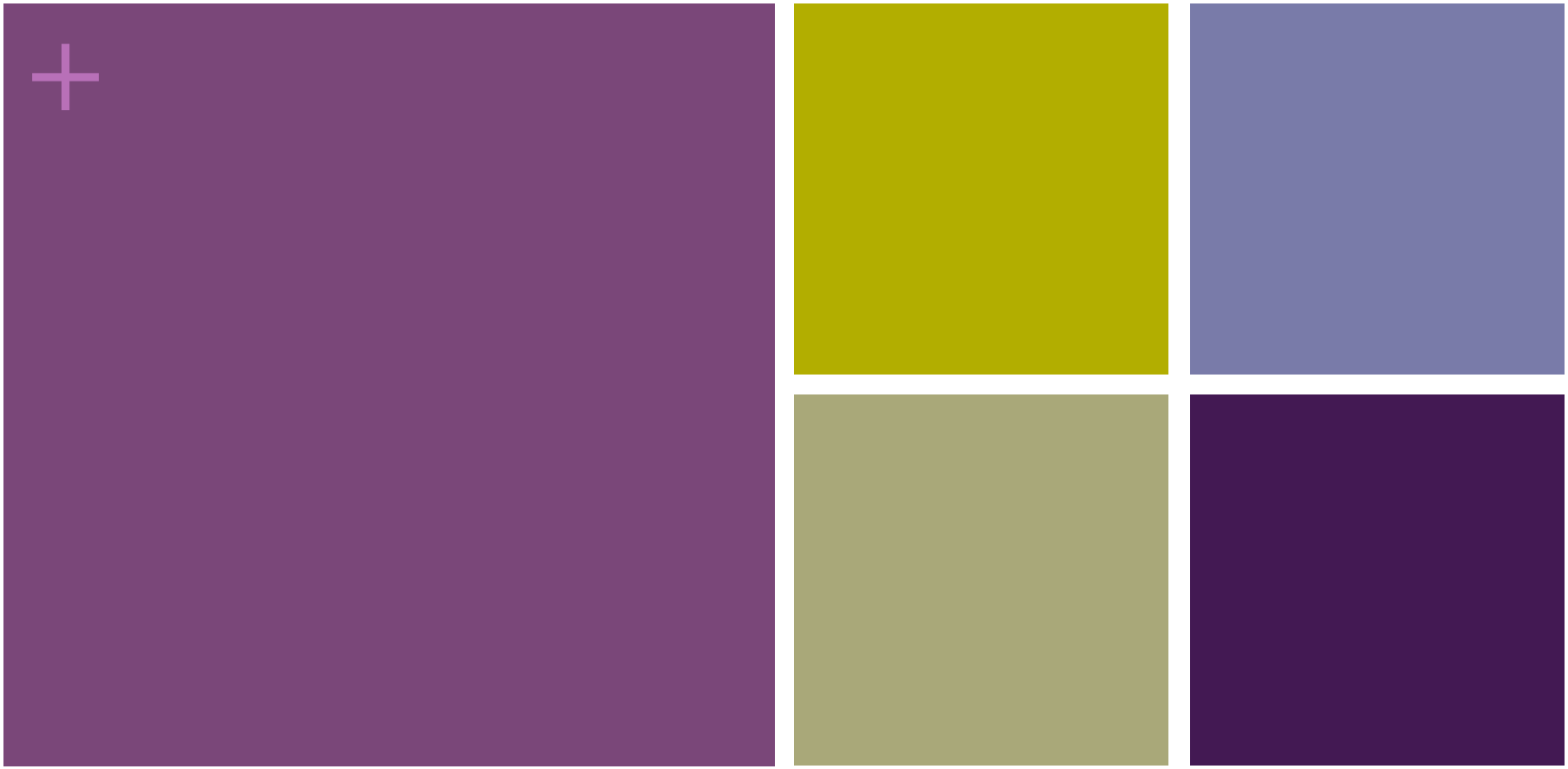


# Database Security

*Lesson 2*

*Database Review*

# Objectives

- Review the basic components of a database
- Define a database and identify the basic components of a database management system
- Identify the different database management system models and applications
- Identify and describe the architecture of Oracle, MySQL, and Microsoft SQL Server

# Database Defined

- Database
  - A collection of data stored on a computer using a database management system (DBMS)
- DBMS (Database Management System)
  - Application that allows others to search stored data
  - Goal: provide means to manipulate, analyze, store, and retrieve information

# Database Structure Components

- Method of storing information in a database
  - Depends on database type
- Common components of digital database management applications
  - Tables
  - Keys
  - Queries

# Tables

- Basic unit of storage within a database
- Represents unique and specific data objects
- Composed of vertical columns and horizontal rows
- Column
  - Also known as field
  - Contains a general category of information with similar data types
- Row
  - Also known as record or tuple
  - Holds distinct units of data

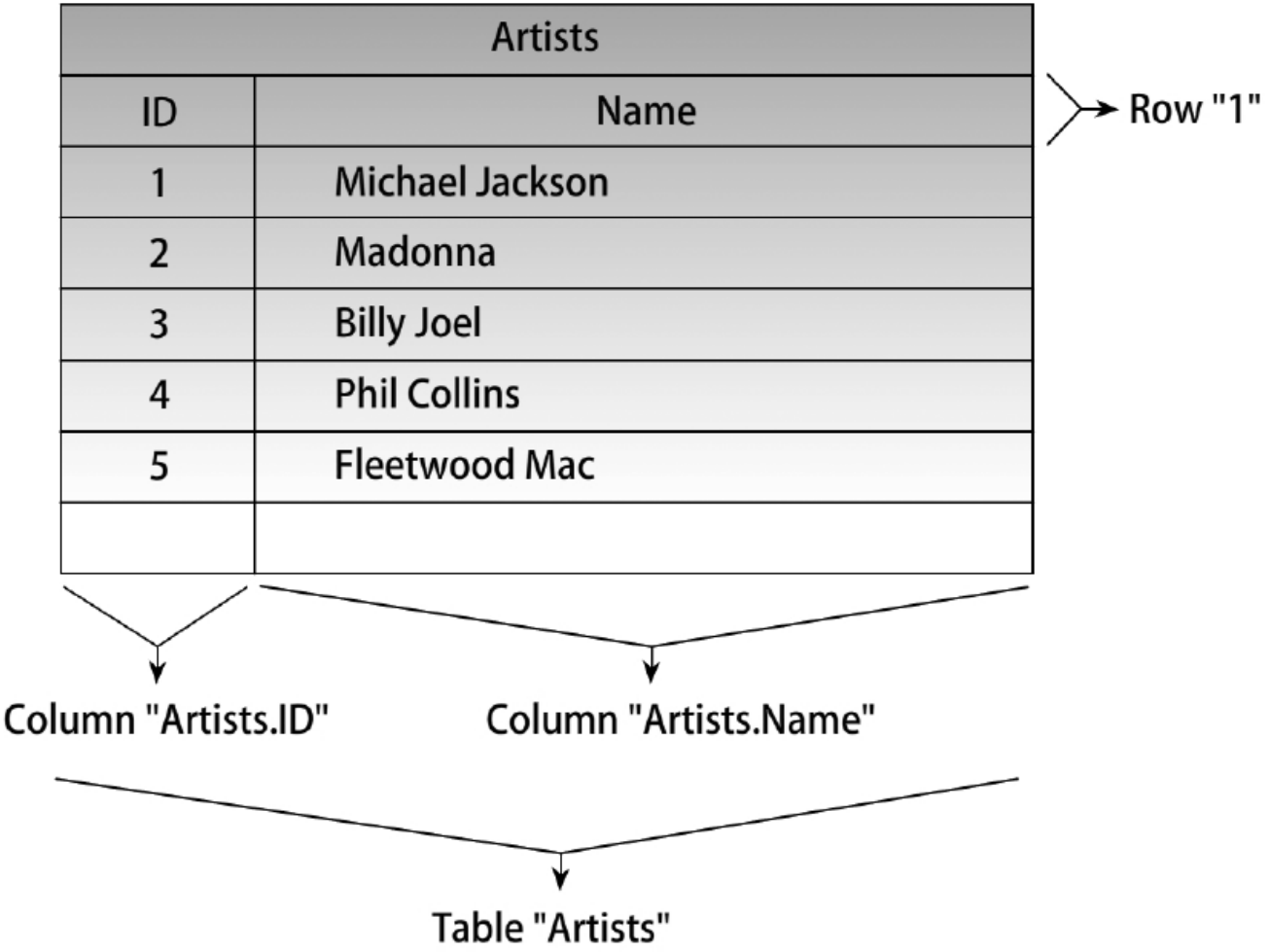


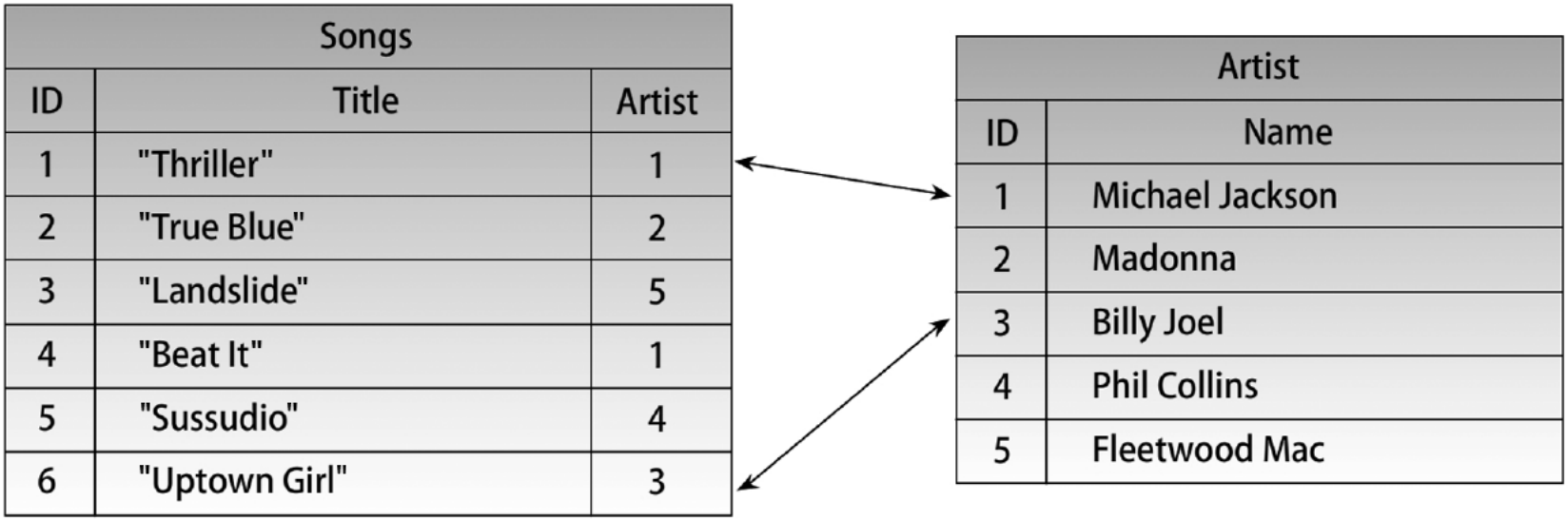

Figure 2-1 Common table found in a database

# Keys

- Single field or group of fields used to identify an entry in a table
- Used to access or manipulate records or rows within a relational database
- Primary key
  - Field containing a unique label identifying a record or row in a table
  - Each table has at least one primary key
  - Keys should be meaningful to the data being stored
  - Key examples: employee ID number, Social Security number

# Keys (cont'd.)

- Foreign key
  - Field within a table containing a label used to build a relationship between two tables
  - Often refers to a unique entry or primary key in a different table
- Other keys
  - Secondary (or alternative) key
  - Candidate key
  - Composite key
  - Sort or control key
  - Alternate key



Songs

| ID | Title | Artist |
|---|---|---|
| 1 | "Thriller" | 1 |
| 2 | "True Blue" | 2 |
| 3 | "Landslide" | 5 |
| 4 | "Beat It" | 1 |
| 5 | "Sussudio" | 4 |
| 6 | "Uptown Girl" | 3 |

Artist

| ID | Name |
|---|---|
| 1 | Michael Jackson |
| 2 | Madonna |
| 3 | Billy Joel |
| 4 | Phil Collins |
| 5 | Fleetwood Mac |

Figure 2-2 Use of foreign keys

# Queries

- Searches initiated by users to retrieve information from the database
- Consist of sets of variables or keywords formatted in a query language
  - SQL is query language used in this text
- Display information in a report

# Structured Query Language (SQL)

- CREATE DATABASE database_name
- CREATE TABLE table_name (column_name1 data_type, column_name2 data_type, column_name3 data_type, …)
- INSERT INTO table_name VALUES (value1, value2, value3,....)
- SELECT column_name(s) FROM table_name
- SELECT * FROM table_name
- SELECT DISTINCT column_name(s) FROM table_name

# Structured Query Language (SQL)

- SELECT column_name(s) FROM table_name1
  UNION
  SELECT column_name(s) FROM table_name2

- SELECT column_name(s)
  FROM table_name
  WHERE column_name operator value

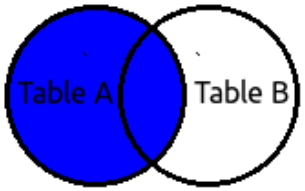
- UPDATE table_name
  SET column1=value, column2=value,…
  WHERE some_column=some_value

- SELECT column_name(s)
  FROM table_name
  ORDER BY column_name [ASC|DESC]

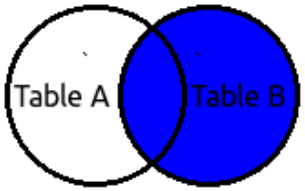
# Structured Query Language (SQL)

- DELETE FROM table_name
- SELECT column_name(s)
  FROM table_name
  WHERE column_name LIKE pattern
- SELECT column_name(s)
  FROM table_name1
  LEFT JOIN table_name2 ON
  table_name1.column_name=table_name2.column_name
- SELECT column_name(s)
  FROM table_name1
  RIGHT JOIN table_name2 ON
  table_name1.column_name=table_name2.column_name

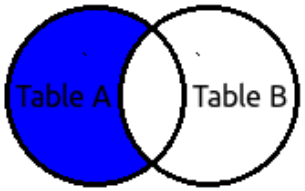
# Structured Query Language (SQL)



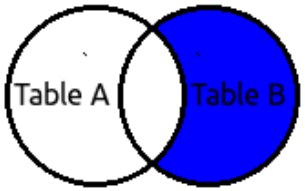

```
SELECT [list] FROM
    [Table A] A
LEFT JOIN
    [Table B] B
ON A.Value = B.Value
```



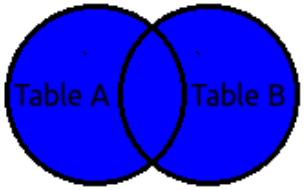

```
SELECT [list] FROM
    [Table A] A
RIGHT JOIN
    [Table B] B
ON A.Value = B.Value
```



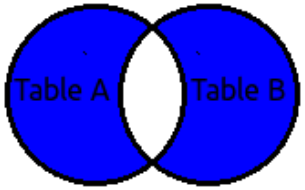

```
SELECT [list] FROM
    [Table A] A
LEFT JOIN
    [Table B] B
ON A.Value = B.Value
WHERE B.Value IS NULL
```



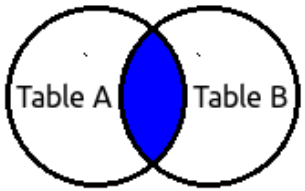

```
SELECT [list] FROM
    [Table A] A
RIGHT JOIN
    [Table B] B
ON A.Value = B.Value
WHERE A.Value IS NULL
```




```
SELECT [list] FROM
    [Table A] A
FULL OUTER JOIN
    [Table B] B
ON A.Value = B.Value
```




```
SELECT [list] FROM
    [Table A] A
FULL OUTER JOIN
    [Table B] B
ON A.Value = B.Value
WHERE A.Value IS NULL
OR B.Value IS NULL
```




```
SELECT [list] FROM
    [Table A] A
INNER JOIN
    [Table B] B
ON A.Value = B.Value
```

# SQL References

- SQL quiz

http://www.w3schools.com/quiztest/quiztest.asp?qtest=SQL

- SQL test

http://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all

- SQL Tutorial

http://www.w3schools.com/sql/default.asp

# Database Models

- Representation of the way data is stored
- Determines how data is retrieved and manipulated
- Four main database models
  - Flat
  - Relational
  - Hierarchical
  - Network

# Flat Model

- Two-dimensional list of data entries
  - All data within a field are similar
  - All data within a record are related to one another
  - Similar to a sign-in sheet at a doctor's office

| Artist | Song |
| --- | --- |
| | |
| | |
| | |

Figure 2-3 Flat database model

# Flat Model (cont'd.)

- Flat model disadvantages
  - Multiple efforts
  - Redundant data
  - System allows large margin for error
    - Entries must be made in exactly the same way or query will not return complete results
    - Example: one user might enter "Billy Joel" and "Uptown girl" and another user "Billie Joel" and Up Town girl"

# Hierarchical Model

- Popular in late 1960s and through the 1970s
- Uses tiers and parent-to-child relationships to represent records and relationships
  - Similar in structure to a family tree
- One-to-many approach greatly minimizes redundancy
- Model builds relationships within one stem
  - No direct relationships made across the tree

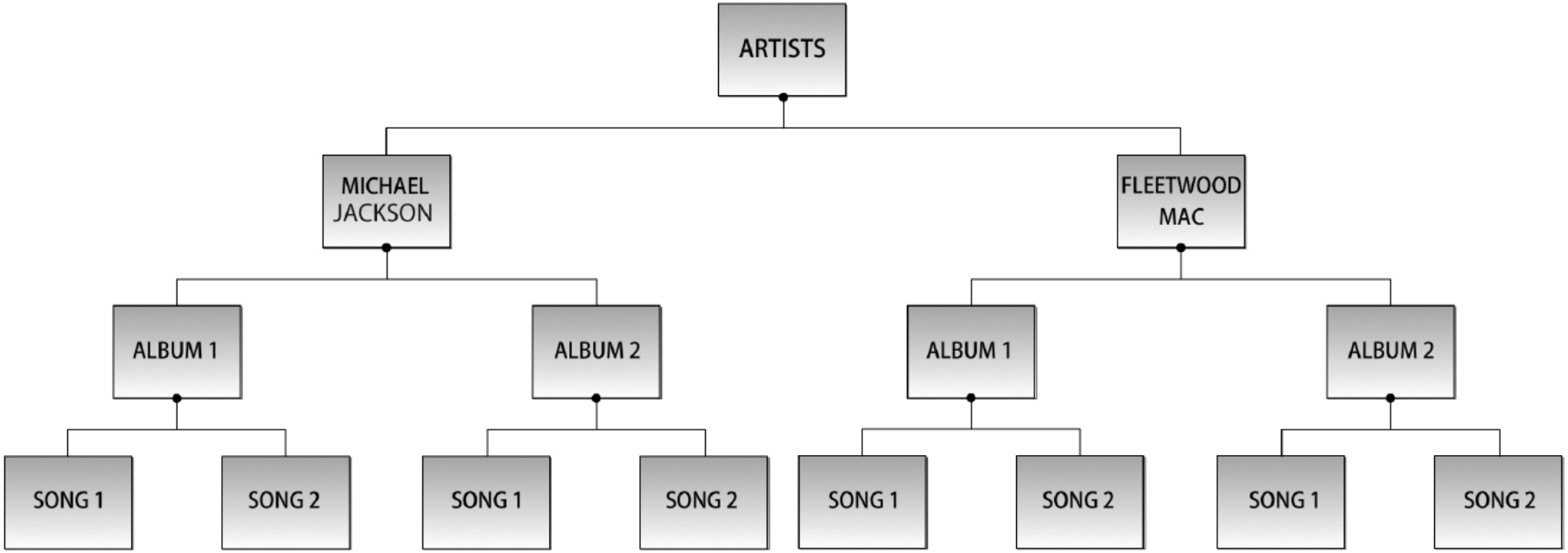



Figure 2-4 Hierarchical database model

# Network Model

- Developed as a solution to one-to-many restrictive nature of hierarchical database model
- Treelike structure using tiers and parent-child-like entities to represent relationships
- Parent referred to as a set of which child entities are members
  - Child entities may be members of more than one set
- Advantages of many-to-many relationship
  - Less resource intensive
  - Easier to navigate

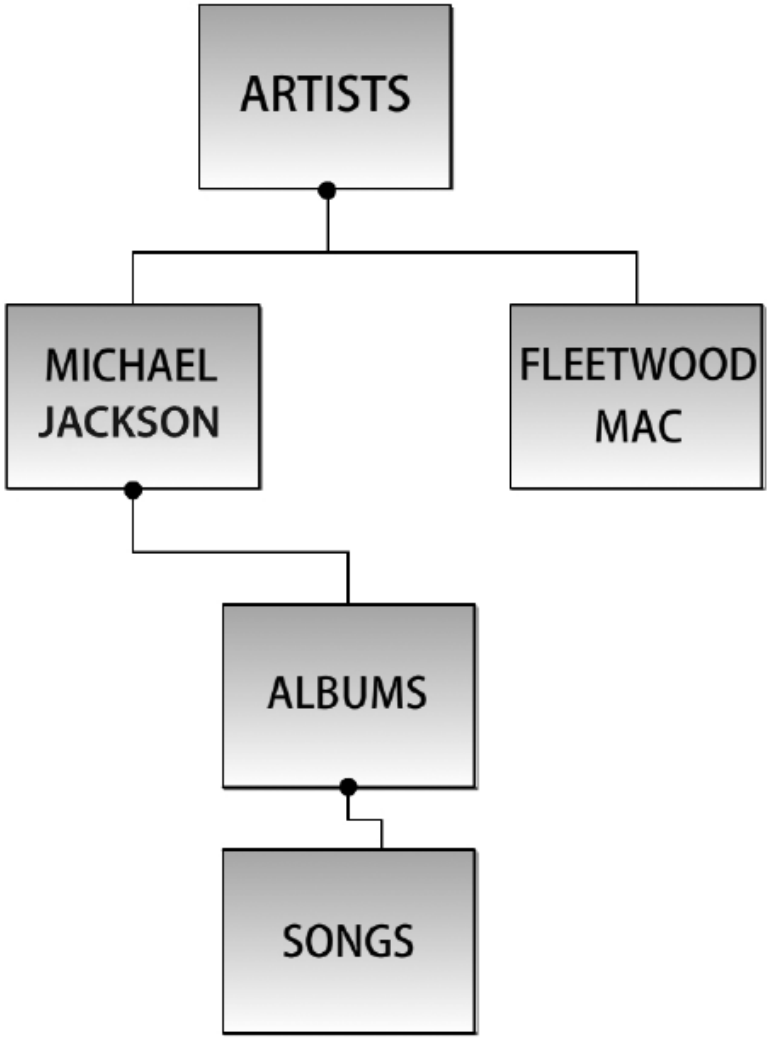
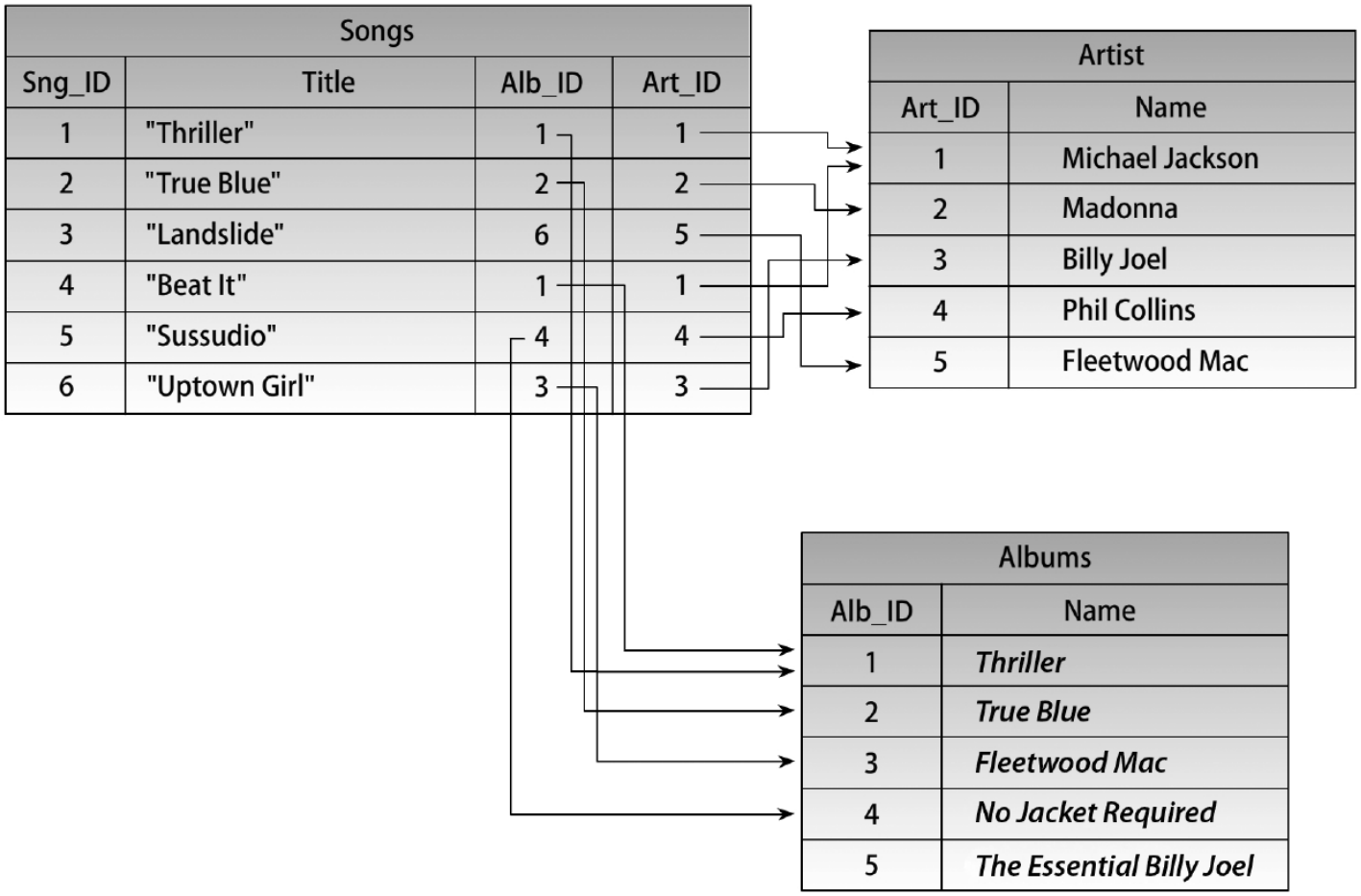



Figure 2-5 Network database model

# Relational Database

- Common entities are stored within separate tables
  - Tables given unique names
  - Tables use unique key identifiers to build relationships among entities
- Entity
  - Person, place, or thing stored in a database table
  - Has attributes and relationships
    - Attribute: describing characteristic
    - Relationship: defines association between two entities



**Songs**

| Sng_ID | Title | Alb_ID | Art_ID |
|---|---|---|---|
| 1 | "Thriller" | 1 | 1 |
| 2 | "True Blue" | 2 | 2 |
| 3 | "Landslide" | 6 | 5 |
| 4 | "Beat It" | 1 | 1 |
| 5 | "Sussudio" | 4 | 4 |
| 6 | "Uptown Girl" | 3 | 3 |

**Artist**

| Art_ID | Name |
|---|---|
| 1 | Michael Jackson |
| 2 | Madonna |
| 3 | Billy Joel |
| 4 | Phil Collins |
| 5 | Fleetwood Mac |

**Albums**

| Alb_ID | Name |
|---|---|
| 1 | *Thriller* |
| 2 | *True Blue* |
| 3 | *Fleetwood Mac* |
| 4 | *No Jacket Required* |
| 5 | *The Essential Billy Joel* |

Figure 2-6 Relational database model

# Object-Oriented Databases

- Object-oriented database management systems
  - Allow storing and retrieving objects and complex data types
    - CAD files
    - Artificial intelligence objects
    - XML-compatible objects
    - General multimedia (audio and video)
  - Were not widely adopted when released in 1985
  - Used in specialized areas today

# Object-Relational Database

- Introduced in the 1990s
- Relational database with expanded group of data types
- Middle ground between relational and object-oriented database management systems
- Not widely used at this time

# Relationships

- Define association between entities and bind them together
- Relationship types
  - One-to-one (1:1)
    - Most simple relationship
    - Rarely found in a relational database
  - One-to-many (1:N)
    - Entity has a sole relationship with entity that has several relationships
  - Many-to-many (M:N)
    - Entity has one or more partnerships with another entity that also has one or many partnerships

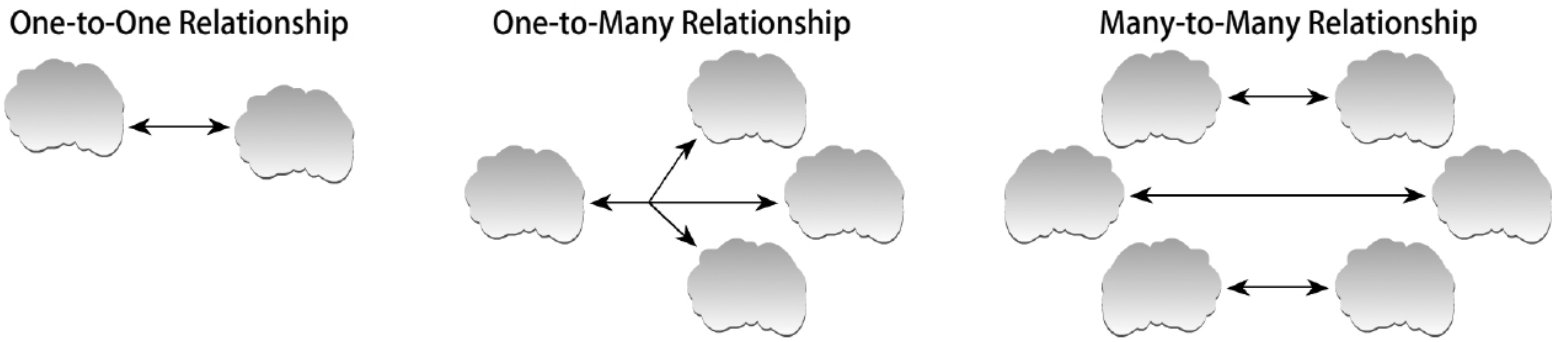



Figure 2-7 Entity relationship types

# Database Types

- Database type is determined by data that will be housed in it
- Types of databases
  - Online transaction processing (OLTP)
  - Online analytical processing or decision support system (OLAP/ DSS)

# OLTP

- Database created for real-time storage and manipulation of data within an organization
- Created to be used in an active environment
- Optimized to serve thousands of users simultaneously
- Stores data resulting from large volumes of short transactions
- Point of sales (POS) system
  - Meant to handle cash register or sales transactions

# OLAP/DSS

- Stores large volumes of historical data
- Used for report generating and analyzing
- Typically retrieves data from an OLTP
- Data analyzed in a business environment to meet a specific need
- Other names
  - Data warehouse
  - Data repository

# Database Management Systems

- Application that provides means to manipulate, analyze, and query data
- Almost all DBMSs existing today are developed to be used with relational databases
  - Known as Relational Database Management Systems (RDBMSs)
- Focuses of this text
  - Oracle
  - Microsoft SQL
  - MySQL

# Oracle

- RDBMS developed by Oracle Corporation in late 1970s
  - Remains a popular database server
- Advantages
  - Portable
  - Can run on almost any operating system
  - Dominant role in providing business solutions
- Current version addressed in this text
  - Oracle 11g

# MySQL

- RDBMS developed by Sun Microsystems
- Most popular open source database server today
- Advantages
  - Speed
  - Open source (available free of charge)
  - Can be customized
  - Platform independent

# Microsoft SQL

- Often referred to as SQL Server
- RDBMS developed by Microsoft
- Primary query languages
  - T-SQL
  - ANSI SQL
- Advantages
  - Scalability
  - Meets needs of any Windows environment

# Database Similarities

- Read consistency
  - Refers to accuracy and reliability of data within a database
  - Depends on database ability to process and commit transactions in a timely manner
  - Applies the following locking mechanisms:
    - Transactions
    - Concurrency
    - Locks
    - Commit
    - Undo

# Database Similarities (cont'd)

- Query Management
  - Steps taken by a database management application to process a user query
  - Retrieving
  - Parsing
    - Analyzing query construction for correct syntax and semantics
  - Optimizing
    - Process of locating most efficient way to retrieve requested data
  - Queries can be processed individually or in parallel

# Oracle Architecture

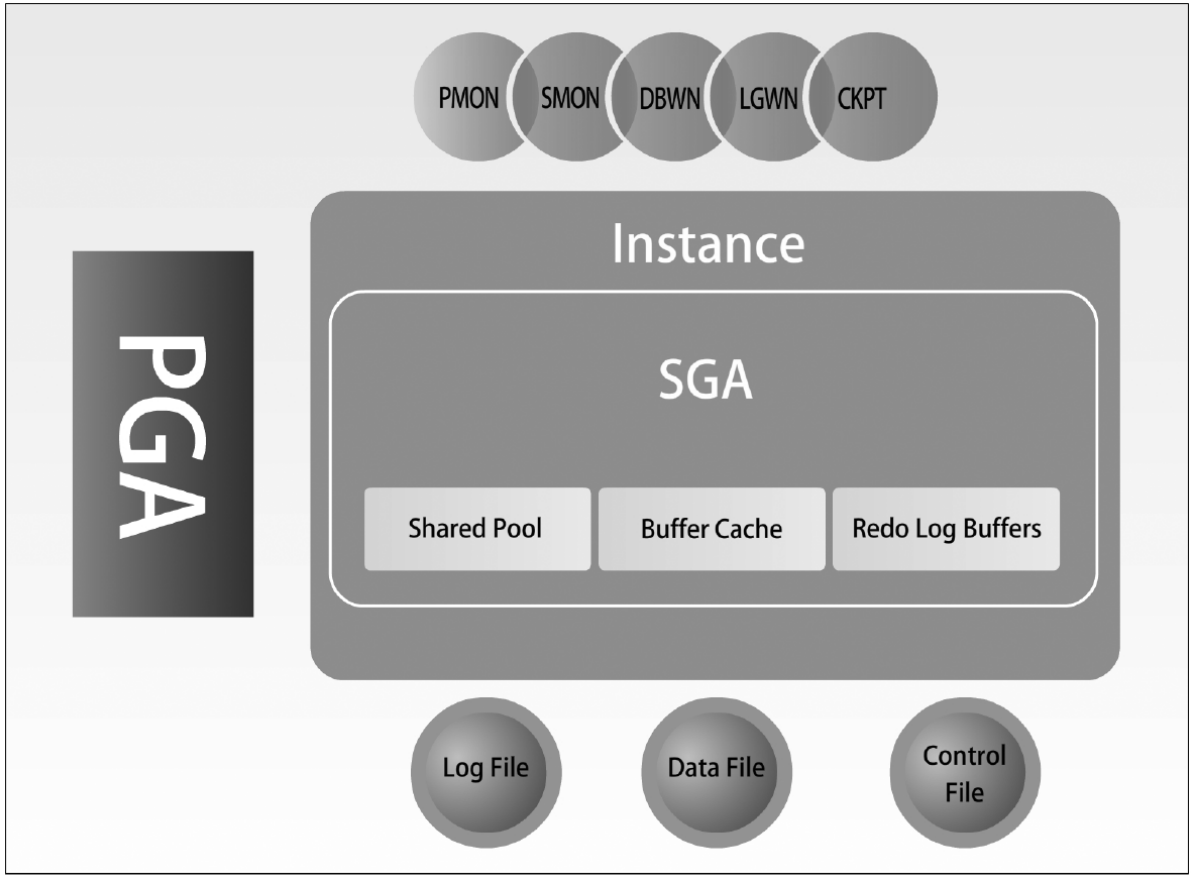



Figure 2-8 Oracle architecture

# The Instance and the Database

- Instance
  - Refers to the background processes and structured memory used during interaction with a database
  - To create, user must connect to a database and establish a session
- Database portion of an Oracle server
  - Holds database files that environment needs to run Oracle database
    - Files help configure the instance, process SQL executions, and ensure alert and recovery from software and hardware failure

# The Physical Structure

- Varies depending on operating system on which Oracle is installed
- Files required with every Oracle install
  - Datafile, control file, and redo log
  - Files interact with OS
  - Transparent to the user
- Datafile
  - Contains actual database data and holds information for all logical structures in the database

# The Physical Structure (cont'd.)

- Control file
  - Contains location and credentialing information of other files
  - Database will not run if control file fails
- Redo log
  - Contains information about all changes made to database data
  - Can be used to restore lost data
  - Good practice to make a duplicate copy of this file

# The Memory Structure

- Main memory and cache
  - Quickest accessible storage in any system
- In Oracle, nearly everything happens from main memory
  - Oracle can reliably service many users concurrently
- Query caching
  - Queries cached into buffer area to increase speed of future query returns
- Oracle memory structure divided into two parts
  - System Global Area and Process Global Area

# The Memory Structure (cont'd.)

- System Global Area (SGA)
  - Central storage area for all shared data and processes
  - Holds control data for one single instance in Oracle
  - Oracle 11g uses dynamic SGA
- Process Global Area (PGA)
  - Central storage area for background and server processes
  - Allocates space for each individual background process
  - Content varies depending on Oracle configuration

Table 2-1 Required memory structures within the SGA

| Structure | Description | Information and examples |
|---|---|---|
| Database buffer cache | Used to cache information read from the data files as well as recently used SQL and PL/SQL queries | A user executes a query on a client machine, and when the client connects to the dedicated server process. The server process first checks the cache to see if the file already exists. If it does, it passes the result to the users; if it is not in the cache, the server process takes data from the datafile and loads it into the buffer cache for future use |
| Shared pool | Stores the most recently executed SQL statements and data definitions Contains the library cache and data dictionary cache | When you execute a SELECT query, once executed, the statement is saved in the shared pool; it checks the rights of the user to ensure the user can view the requested information and then removes the requested information from the datafile, loading it into the buffer cache for the user to view |
| Library cache | Used to cache metadata information | An SQL statement is stored here parsed while the syntax is checked by the database; once validity is confirmed, it searches in the shared pool for a cached version of the statement |
| Data dictionary cache | Caches recently used data dictionary information | User account information, datafile names, etc. |
| Database buffer cache | Stores blocks of table and database data that has been retrieved in the past | A query is sent to the server process; the server process first looks in the database buffer to find the information needed to access the hard disk, thus speeding up performance |
| Redo log buffer | Stores all changes that have been made to the database | When the redo buffer gets filled, the excess moves to the redo file |

| Structure | Description | Information/example |
|---|---|---|
| Large pool | Stores large jobs to avoid filling the shared pool | RMAN backups |
| Java pool | Stores and caches Java commands | Necessary only when Java is installed |
| Streams pool | For advanced queuing | Only used in Oracle 10g and 11g |

Table 2-2 Optional memory structures within the SGA

# The Processes

- Process
  - Instruction set executed by OS to complete a task
- Tasks required to complete an instance in Oracle
  - User runs application tool to request connection to Oracle server (user process)
  - Server handles user's request and runs process to create instance and complete the connection (server process)
- Server processes can be shared or dedicated

| Process name | Example process task |
|---|---|
| PMON (process monitor) | Cleans up after processes complete or processes fail |
| SMON (system monitor) | Recovers the database in case of failure by using the redo logs as well as database files |
| DBWN (database writer) | Writes changes from the database buffers to the database files |
| LGWR (log writer) | Writes modifications from the redo log buffer to the redo log files |
| CKPT (checkpoint) | Writes to the control file's established commit points where recovery begins, if necessary |

Table 2-3 Background processes

# MySQL Architecture

- Developed for multiplatform use
  - Solaris, Linux, Windows
- Architecture components common to these operating systems
  - Database connection manager
  - Query engine
  - Transaction manager
  - Storage engine

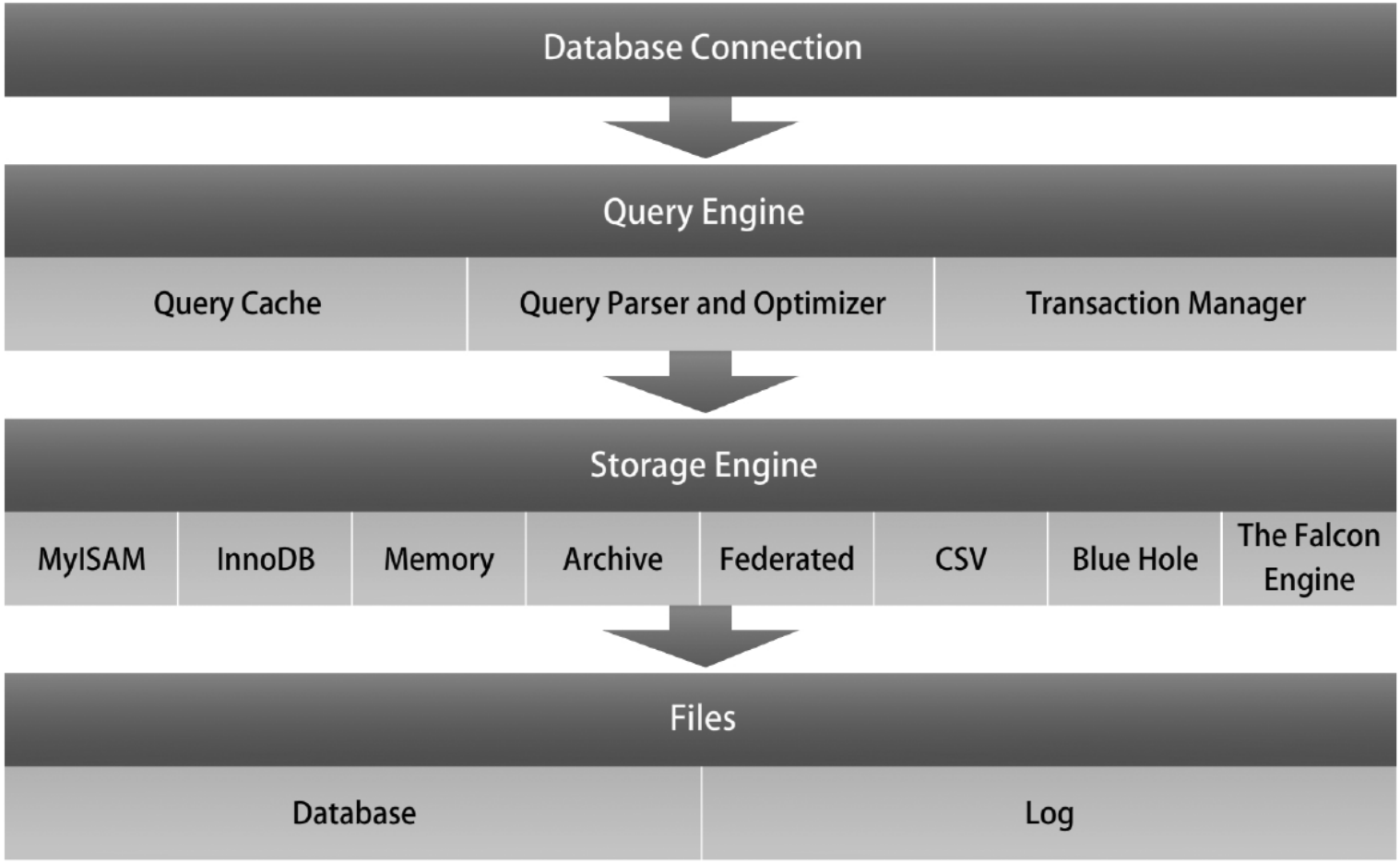



Figure 2-9 MySQL architecture

# Database Connection Manager

- Manages connections to the MySQL server
  - Virtually any client may connect
- Methods for clients to connect via the connection management layer
  - Application programming interfaces (APIs)
    - Many programming languages are supported
    - Clients using ODBC, Java, .net are supported
  - TCP/IP is most common type of connection
- Thread
  - Execution running independently from other processes

# Query Engine

- Architecture component that optimizes and manages queries and SQL statements
- Built to use resources efficiently
- Query steps
  - User initiates query
  - Query request received by MySQL server
  - Query parser creates treelike structure of extracted SQL statements
  - Data definition language provides access
- Memory cache stores recently requested queries

# Transaction Manager

- MySQL transaction
  - Group of MySQL queries treated as a single process
- Transaction manager maintains concurrency throughout the database
  - Ensures simultaneous data handling will not corrupt data
- Types of database environments
  - Transactional
  - Nontransactional

| ACID characteristic | Description | MySQL sample compliance |
|---|---|---|
| Atomicity | SQL statements operate together as one entity group or alone as one entity; a group passes and/or fails as one process | MySQL statements begin and end, as well as pass and fail, together using the BEGIN and COMMIT, UNDO, and ROLLBACK statements |
| Consistency | Transactions do not affect the state of the database; consistency remains despite the success or failure of a transaction | MySQL uses logs (binary logs) to record all changes to the database as well as to help recover from a failure; the ROLLBACK statement is used if needed |
| Isolation | Transactions run separated from other transactions and are not viewable until committed | MyISAM permits locking, which avoids data corruption and visibility |
| Durability | Transactions persist despite system failures | Binary logs can be reverted |

Table 2-4 ACID compliance and MySQL

# Storage Management

- MySQL stores data files in secondary storage
  - Dynamically traversing files is too slow
- Storage management
  - Process of storing and retrieving data throughout the database
  - Most work takes place within main memory
  - Three-tiered process uses resource manager, buffer manager, and storage manager

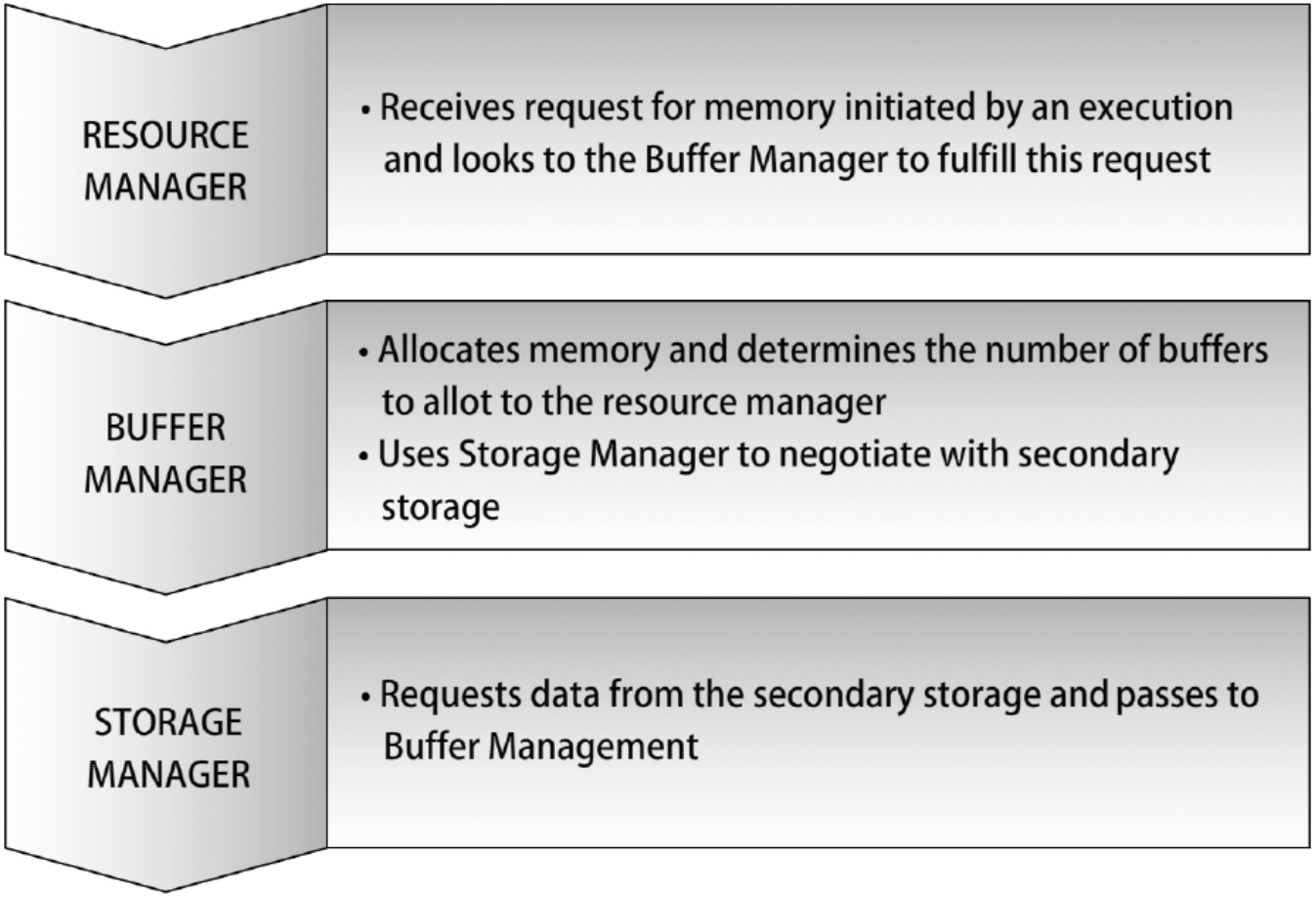



Figure 2-10 MySQL storage management

# The Storage Engine

- Components that read and write data to and from the database
- Customization options available
  - Administrators can choose which storage engines to use for certain tables or applications
- Storage engines available in MySQL
  - MyISAM, InnoDB, Memory, Archive, Federated, Comma-separated values, Black Hole Engine, The Falcon Engine

# Microsoft SQL Server Architecture

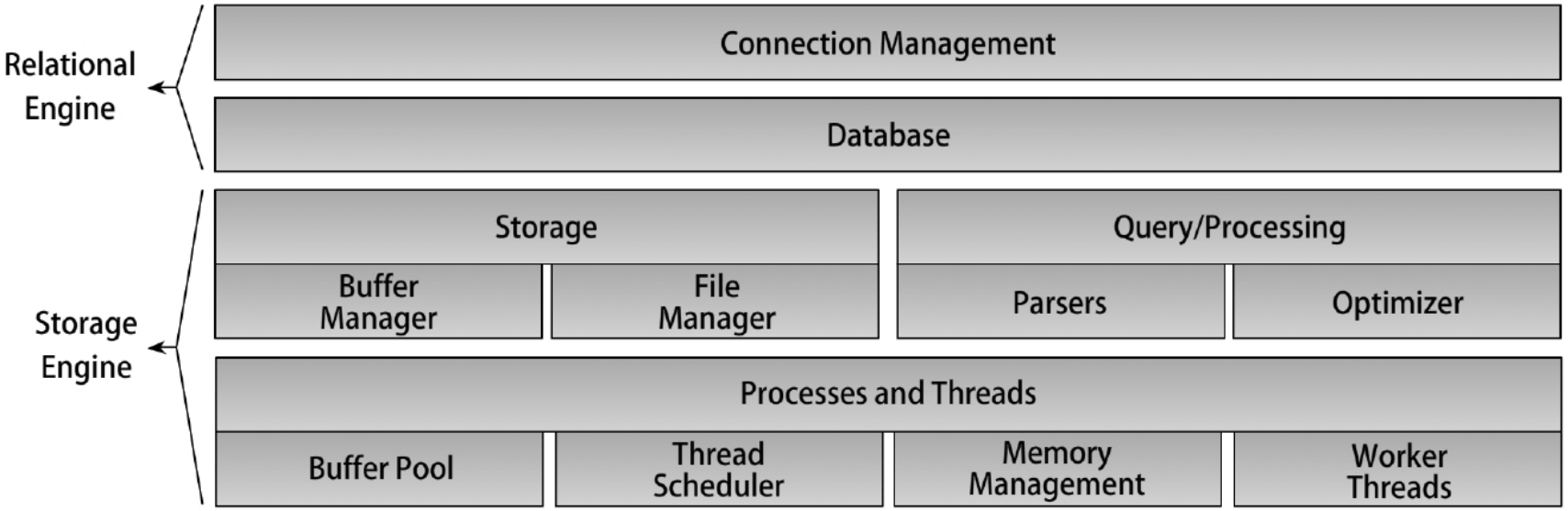



Figure 2-11 Microsoft SQL Server architecture

# Architecture and Engines

- Clients connect to the server using the Tabular Data Stream
- Main components of SQL Server DBMS architecture
  - The relational engine
    - Responsible for query processing and data retrieval
  - The storage engine
    - Manages files, memory, recovery, logging, and transactions

# The Physical Structure

- Three different types of files store data
  - Primary data files
  - Secondary data files
  - Log files
- Each installation of SQL server
  - Has at least one primary file and one log file
- Filegroups used to organize database into logical units of resources
  - Primary filegroup
  - User-defined filegroup

# Memory Management

- SQL Server can dynamically allocate its own memory
  - No need for administrator intervention
  - Balances main memory use between database processes and storing and retrieving data
- Virtual memory
  - Technique for extending memory availability
  - Units of storage from different memory devices appear as a single block of storage
  - Fixed units of storage referred to as pages

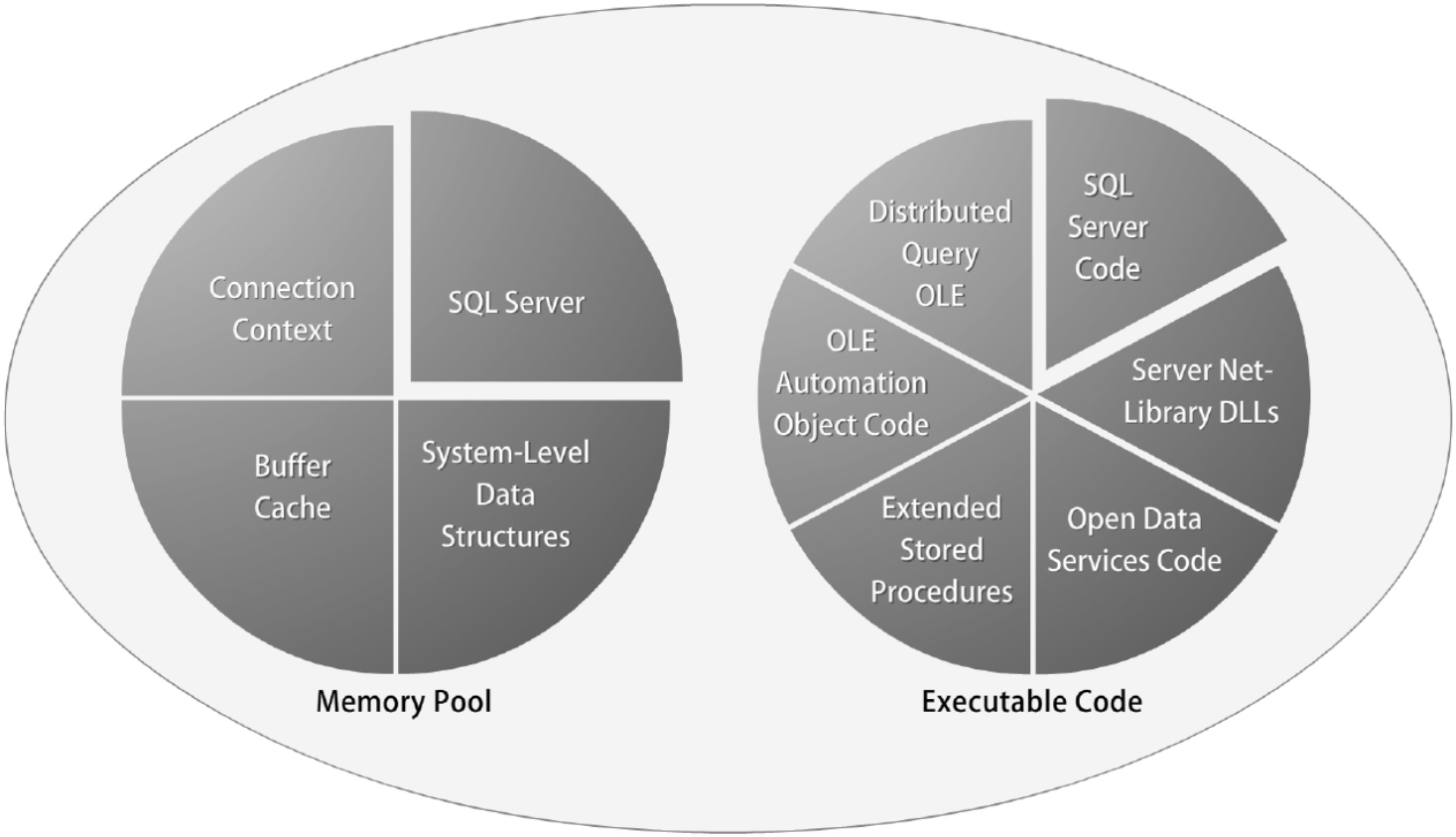



Figure 2-12 Microsoft SQL Server memory management

# Buffer Management

- Buffer manager accesses data pages and updates the database
- Buffer pool (buffer cache)
  - Area in which data pages are stored
  - Minimizes need to read and write from hard disk file
- Steps to retrieve data after query processing
  - Buffer manager accesses database files from hard disk and places in buffer cache pages
  - Data read from the cache
  - Page considered dirty if changes made to data while in buffer cache

# Threads and Processes

- Threads and fibers used to perform several tasks simultaneously
  - Threads handled by the operating system and allocated one per CPU
  - Fibers handled by the server and allocated one per user command
- Worker processes
  - Pools of either threads or fibers for all user connections
  - Number of threads or fibers within one worker process available depends on network size

# Summary

- A database allows others to search, analyze, and manipulate stored data
- Data is stored within a database using a table structure
- Several types of keys are used to reference data
- Query languages provide users a way to retrieve and manipulate data from a database
  - SQL is most common query language
- General query processing steps
  - Parsing, optimization, execution of SQL statements

# Summary (cont'd.)

- Different database models describe how data is stored within a database
  - Flat, hierarchical, network, and relational
- One-to-many relationships are most common relationships formed between tables in a relational database
- Most common DBMSs include Oracle, MySQL, and Microsoft SQL Server
- Read consistency and query management are important aspects of a DBMS