



Introduction to Database Management



www.its.ac.id



[its_campus](#)



[institut teknologi sepuluh nopember](#)



About the Instructor

Shintami Chusnul Hidayati, Ph.D.

Department of Informatics, ITS



shintami@its.ac.id



Database Management (EF234404)

3 credits



About the Course

IUP

Monday, 13:30-15:20
TIF-108

E

Thursday, 10:00-11:50
TIF-103

F

Thursday, 13:30-15:20
TIF-104



About the TA



XYZ

502 ■ ■ ■ ■ ■ ■



Students will learn about the **modeling of complex systems** in industry based on business processes. According to the reference model, students will **implement and manage an optimal SQL database**. Lectures are presented in the classroom, and students will work on small projects as a practice. This course aims to provide experience to students in managing and handling problems when **working with large-scale data**. This course will cover **distributed databases** and **data warehouses** as well.



Learning Outcome

- Students can **model database** from various industrial fields.
- Students can handle the problem in a **large-scale database**.
- Students can model an **active database** integrated with business rules.



Topics

- The relational model and relational algebra
- SQL (the Structured Query Model)
- The Entity-Relationship model
- Database schema design and normal forms
- Various common uses of database systems



Week-by-Week List of Topics

Week I	(1) Explanation of the course syllabus; (2) Review of database and data modeling: database concept,
Week II	entity, attribute, data modeling on CDM and PDM, information elicitation using a simple query; (3) Explanation of Database Management benefits in real life
Week III	Nested query
Week IV	Complex Query
Week V	Competence Test 1
Week VI	(1) Active database and trigger: passive and active database concept; (2) Generic model of active database and trigger; (3) Active database applications; (4) Active database design and implementation issues
Week VII	(1) Active database applications; (2) Integrity management; (3) Derived data maintenance; (4) Workflow management; Business rule; (5) Transaction processing
Week VIII	Mid-term Exam

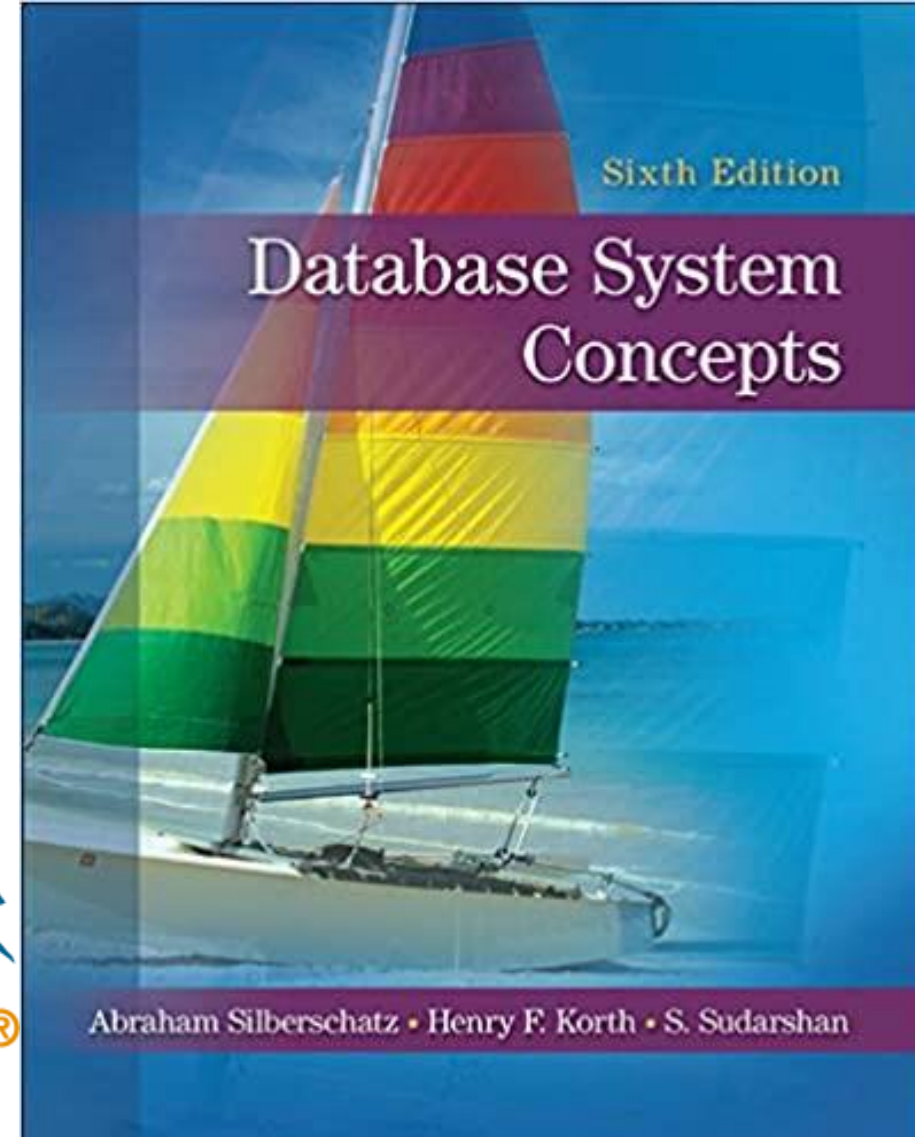


Week-by-Week List of Topics

Week IX	(1) Indexing: Indexing concept and function; Instruction on index selection; examples of index usage; Clustering and indexing; Index which enables index planning only. (2) Tools for index selection(automatic index selection)
Week X	Query optimization and manipulation
Week XI	Parallel database: parallel database architecture, parallel query, parallelizing individual operations, parallel query optimization
Week XII	Database Administrator: replication and security, backup and recovery
Week XIII	Database Tuning
Week XIV	Competence Test 2
Week XV	Recap
Week XVI	Final Project Demo



Textbook/Tool





Assignments

- Assignments are given approximately weekly
 - Set of problems focusing on that week's material
 - Made available during the class
 - Due six days later
- Assignment and exam weighting:
 - Competence Test 1: 15%
 - Midterm Exam: 30%
 - Competence Test 2: 20%
 - Final Project Demo: 35%



Grading Policies

- Late assignments and exams will be penalized!
 - Up to 1 day (24 hours) late: 10% penalty.
 - Up to 2 days (48 hours) late: 20% penalty.
 - Up to 3 days (72 hours) late: 30% penalty.
 - After 7 days, don't bother.
- But, extensions are available:
 - A note from Dean's Office or Health Center is helpful
 - You have 1 "late token" to use however you want
 - Each late token is worth a 72-hour extension
 - Can't use late tokens on the midterm and final exam without my permission





Database Terminology

- Database – an organized collection of information
 - A very generic term...
 - Covers flat text-files with simple records...
 - ...all the way up to multi-TB data warehouses!
 - Some means to query this set of data as a unit, and usually some way to update it as well
- Database Management System (DBMS)
 - Software that manages databases
 - Create, modify, query, backup/restore, etc.
 - Sometimes just “database system”



Database Management System (DBMS)

- Database Applications:
 - Banking: transactions
 - Airlines: reservations, schedules
 - Sales: customers, products, purchases
 - Online retailers: order tracking, customized recommendations
 - Manufacturing: production, inventory, orders, supply chain
 - Human resources: employee records, salaries, tax deductions
- Databases can be very large
- Databases touch all aspects of our lives



University Database Example

- Application program examples
 - Add new students, instructors, and courses
 - Register students for courses, and generate class rosters
 - Assign grades to students, compute grade point averages (GPA) and generate transcripts
- In the early days, database applications were built directly on top of file systems



Before DBMSes Existed...

- Typical approach:
 - Ad-hoc or purpose-built data files
 - Special-built programs implemented various operations against the database
- Want to perform new operations?
 - Create new programs to manipulate the data files!
- Want to change the data model?
 - Update all the programs that access the data!
- How to implement transactions? Security? Integrity constraints?



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



Drawbacks of using File Systems to Store Data (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 access needed for performance
 - Uncontrolled concurrent accesses can lead to inconsistencies
 - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- Security problems
 - Hard to provide user access to some, but not all, data



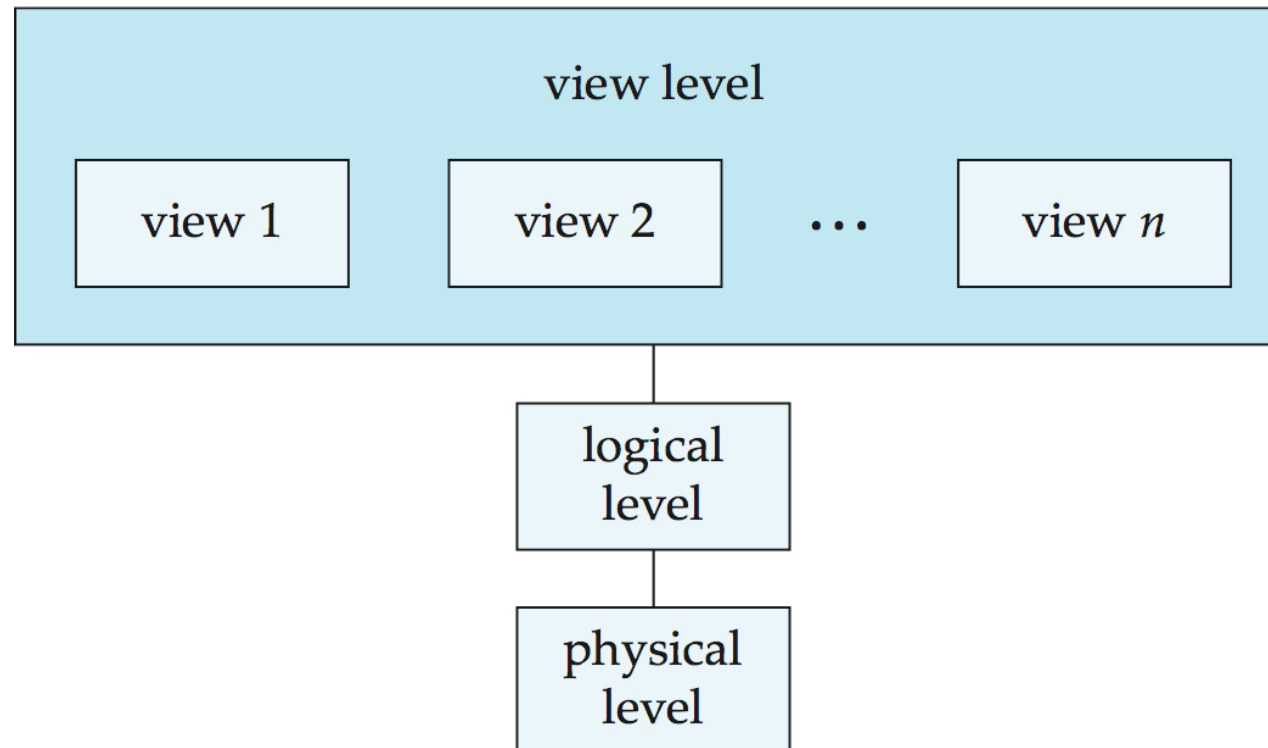
Enter the DBMS

- Provide layers of abstraction to isolate users, developers from database implementation
 - **Physical level**: how values are stored/managed on disk
 - **Logical level**: specification of records and fields
 - **View level**: queries and operations that users can perform (typically through applications)
- Provide general-purpose database capabilities that specific applications can utilize
 - Specification of database schemas
 - Mechanism for querying and manipulating records



View of Data

An architecture for a database system





Kinds of Databases

- Many kinds of databases, based on usage
- Amount of data being managed
 - **Embedded databases**: small, application-specific systems (e.g. SQLite, BerkeleyDB)
 - **Data warehousing**: vast quantities of data (e.g. Oracle)
- Type/frequency of operations being performed
 - **OLTP**: Online Transaction Processing
 - “Transaction-oriented” operations like buying a product or booking an airline flight
 - **OLAP**: Online Analytical Processing
 - Storage and analysis of very large amounts of data
 - e.g. “What are my top selling products in each sales region?”



Instances and Schemas

- Similar to types and variables in programming languages
- **Logical Schema** – the overall logical structure of the database
 - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
 - ▶ Analogous to type information of a variable in a program
- **Physical schema** – the overall physical structure of the database
- **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



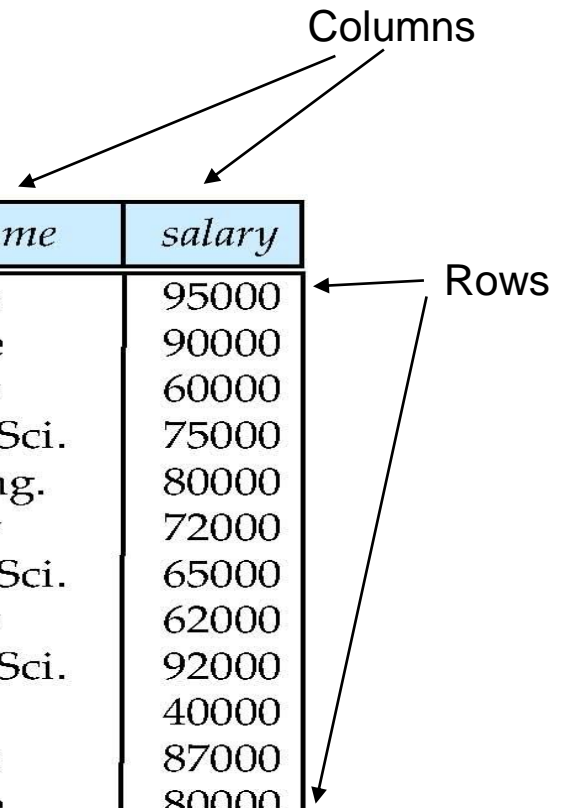
Data Models

- 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



Relational Model

- All the data is stored in various tables
- Example of tabular data in the relational model



<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table



A Sample Relational Database

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

<i>dept_name</i>	<i>building</i>	<i>budget</i>
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table



Data Definition Language (DDL)

- Specification notation for defining the database schema

Example: **create table** *instructor* (
 ID **char** (5) ,
 name **varchar** (20) ,
 dept_name **varchar** (20) ,
 salary **numeric** (8,2))

- DDL compiler generates a set of table templates stored in a *data dictionary*
- Data dictionary contains metadata (i.e., data about data)
 - Database schema
 - Integrity constraints
 - Primary key (ID uniquely identifies instructors)
 - Authorization
 - Who can access what



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
 - **Pure** – used for proving properties about computational power and for optimization
 - Relational Algebra
 - Tuple relational calculus
 - Domain relational calculus
 - **Commercial** – used in commercial systems
 - SQL is the most widely used commercial language



Structured query language (SQL)

- The most widely used commercial language
- SQL is NOT a Turing machine equivalent language
- To be able to compute complex functions SQL is usually embedded in some higher-level language
- 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



Database Design (Cont.)

- Is there any problem with this relation?

<i>ID</i>	<i>name</i>	<i>salary</i>	<i>dept_name</i>	<i>building</i>	<i>budget</i>
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000



Design Approaches

- Need to come up with a methodology to ensure that each of the relations in the database is “good”
- Two ways of doing so:
 - Entity Relationship Model
 - Models an enterprise as a collection of *entities* and *relationships*
 - Represented diagrammatically by an *entity-relationship diagram*
 - Normalization Theory
 - Formalize what designs are bad, and test for them



Object-Relational Data Models

- Relational model: flat, “atomic” values
- 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



Database Engine

- Storage manager
- Query processing
- Transaction manager



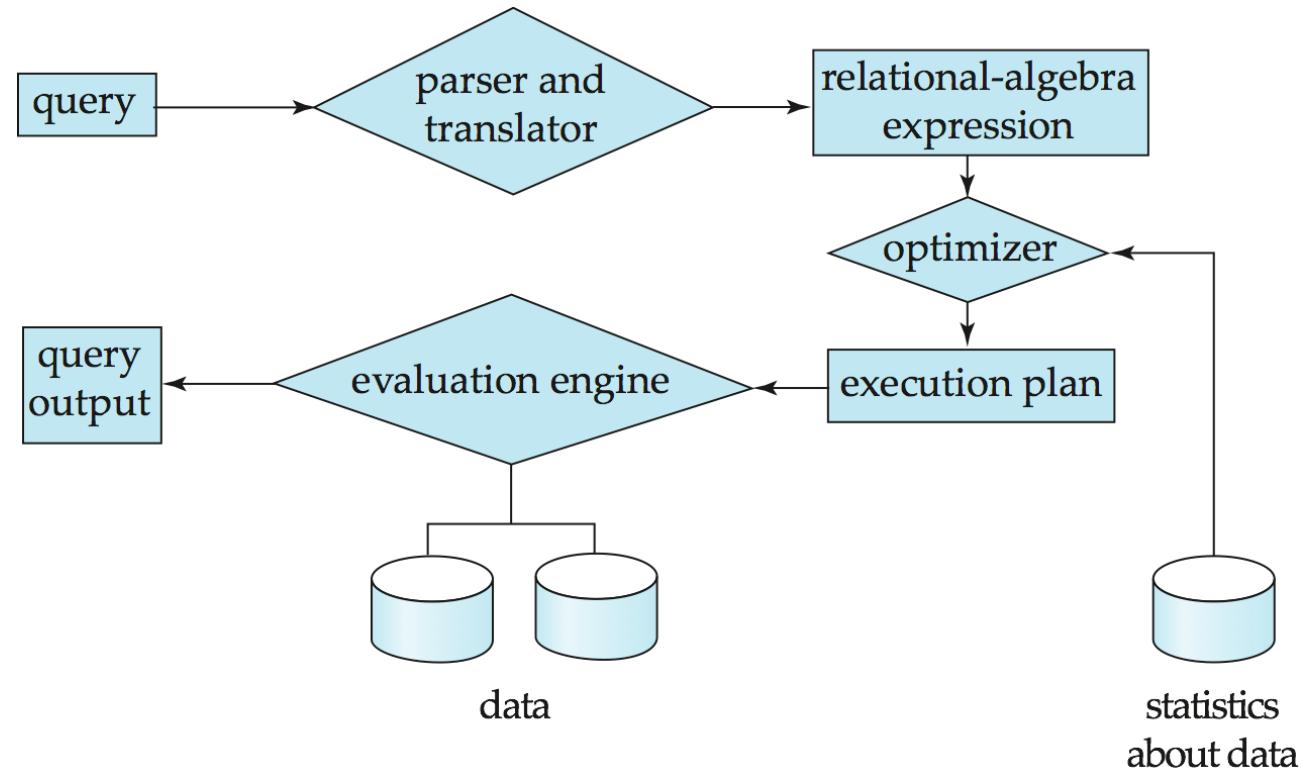
Storage Management

- **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 OS 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

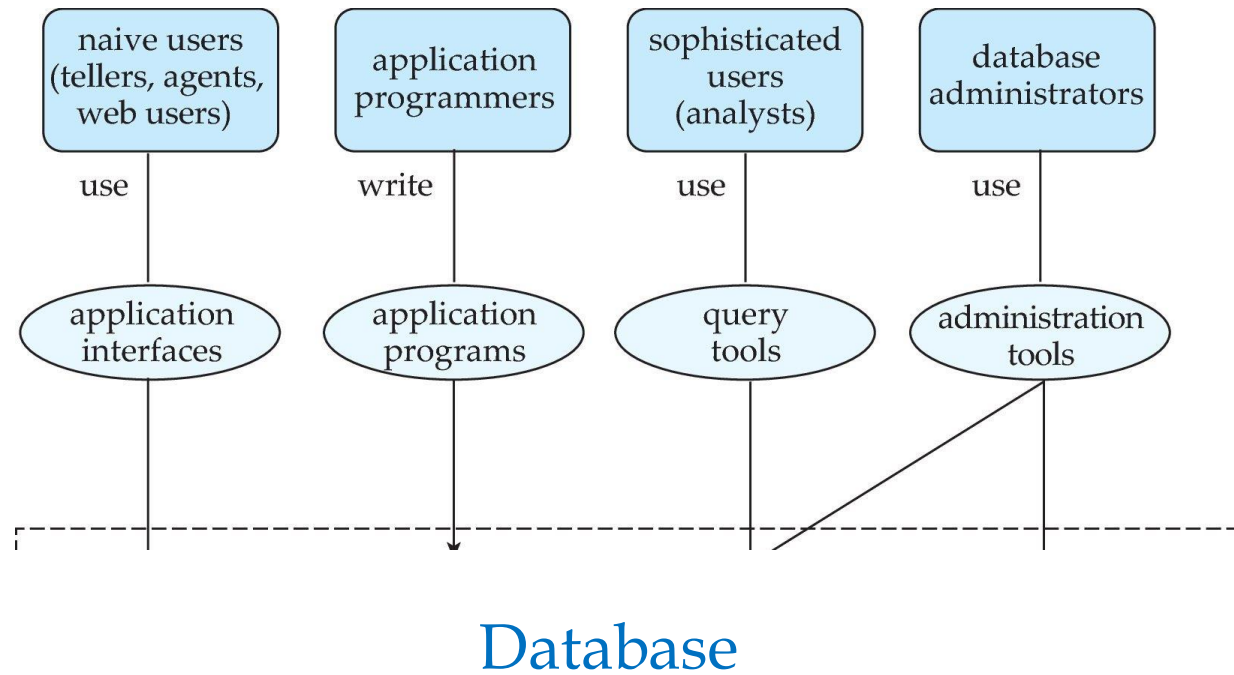


Transaction Management

- What if the system fails?
- What if more than one user is concurrently updating the same data?
- 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.

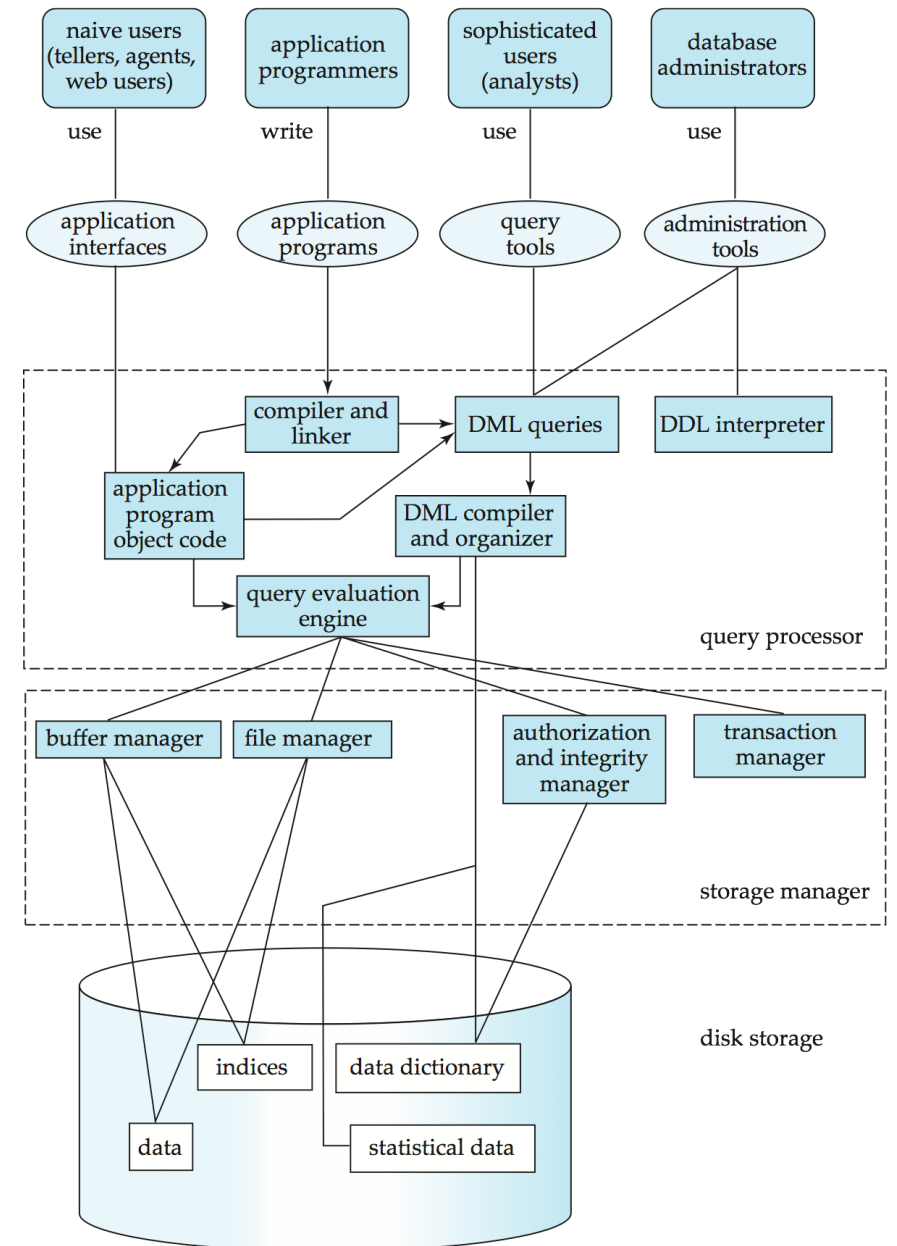


Database Users and Administrators





Database System Internals





Thank You!