COMS W4111: Introduction to Databases — Week 1 Notes

Class Meeting Date: September 10, 2026 (Week 1)
Course Name: COMS W4111 (Introduction to Databases)
Instructor: Prof. Oktie Hassanzadeh
Note-Taker: Panfeng Jiang

Course Administration & Important Announcements

- **Course Platform:** All course materials, syllabus, and assignment portals live at `dbcourse.app` (sign in with Columbia Google account).
- **HW1 (Domain & Conceptual Model):** Opens September 17, 2026.
- **Quiz (How This Course Works):** Due September 17, 2026.
- **Teaching Staff:** Prof. Oktie Hassanzadeh and 4 TAs. Contact early for help as material compounds weekly.

1. What Is a Database Management System (DBMS)?

A DBMS is a software system that manages interrelated data and sits between applications and physical storage. It provides a convenient and efficient environment for data definition, manipulation, integrity, and administration.

Why Not Just Use Plain Files?

Before database systems, applications managed raw files directly. Plain file storage creates major failure modes:

- **Data Redundancy & Inconsistency:** The same fact is stored in several files/formats, causing data to drift apart.
- **Difficulty Accessing Data:** Every new question requires writing a new program or parser.
- **Integrity & Atomicity Problems:** Business constraints are hardcoded in application logic; mid-operation crashes leave data corrupted.
- **Concurrent-Access Anomalies:** Simultaneous updates cause lost writes (e.g., two concurrent withdrawals).
- **Security Issues:** No fine-grained access control at the record or column level.

Concern	Files + Application Code	DBMS Solution
Schema	Implicit in application code	Declared and enforced
Querying	Write new code per question	Declarative SQL language
Integrity	Re-implemented per app	Declared once, enforced for all writes
Concurrency	Ad hoc file locking	Transaction-based isolation
Crash Recovery	Manual / unreliable	Automated log-based recovery
Access Control	File-level all-or-nothing	Fine-grained (per table, column, row)

→ **Data Independence:** Applications state *what* data they want rather than *how* to retrieve it. This allows the DBMS to alter physical layouts (e.g., adding indexes or reordering blocks) without breaking existing queries.

2. Categories of Data & Data Models

Data architecture is driven primarily by **Variety** (one of the 5 Vs: Volume, Velocity, Variety, Veracity, Value).

1. **Structured:** Fixed schema defined prior to data arrival (e.g., Relational tables).
2. **Semi-Structured:** Self-describing; structure travels with each record (e.g., JSON documents, XML).
3. **Unstructured:** Lacks internal structure usable for direct database querying (e.g., Free text, images, audio).

→ **Data Model:** A collection of conceptual tools for describing data, relationships, semantics, and constraints.

- **Entity-Relationship (ER) Model:** A design notation used to reason about domain logic before building (translated to relational tables; no production DBMS implements pure ER directly).
- **Relational Model:** Data organized in tables; the foundational model for relational database systems.

3. The Four Phases of Database Design

Database design moves through four structured phases:

1. **Requirements Analysis:** Identifies user data needs → Characterization of data needs.
2. **Conceptual Design:** Establishes entities, relationships, and constraints → Model-independent ER diagram.
3. **Logical Design:** Translates ER model into relations and attributes → Relational schema.
4. **Physical Design:** Determines physical storage, partitioning, and indexes → Index & storage specs.

Two Distinct Decisions inside Logical Design

Logical design combines two fundamentally separate choices:

- **Business Decision:** Which facts about the enterprise should be recorded at all?
- **Computer Science Decision:** How should those facts be grouped into tables? (Normalization).

→ **Physical Design Is Deliberately Last:** Index selection and storage parameters must respond to measured workload evidence rather than speculative guesses.

4. Relational Concepts & Terminology

In the relational model (proposed by Ted Codd), all data lives in mathematical tables.

- **Relation:** A table. Rows are **tuples**, columns are **attributes**.
- **Domain:** The set of permitted values for an attribute.
- **Atomic Domains:** Attribute values must be indivisible. Storing non-atomic composite values (e.g., `'Ferguson, Donald, F'` or `'COMSW4111'`) forces every query to parse strings, introducing bugs.
- **Unordered Set:** Tuples in a relation have no default physical order.
- **Schema vs. Instance:** Schema is the logical structure (e.g., `instructor(ID, name, dept_name, salary)`); Instance is a snapshot of data at a specific moment in time.

5. Key Definitions

Keys express tuple identity and establish relationships across relations.

- **Superkey:** A set of attributes $K \subseteq R$ sufficient to uniquely identify a tuple in *every* legal instance of R.
- **Candidate Key:** A minimal superkey (no proper subset is also a superkey).
- **Primary Key:** The candidate key chosen by database designers to identify tuples internally (prefer immutable keys).
- **Foreign Key:** Attributes in a referencing relation whose values must match a candidate/primary key in a referenced relation. Enforces Referential Integrity and prevents dangling references.

Foreign Key Rules & Deletion Policies

- **Enforcement:** Checked automatically on every write operation (`INSERT`, `UPDATE`, `DELETE`).
- **NULL Values:** Foreign keys may be `NULL` (represents the absence of a reference, not a violation).
- **Referenced Row Deletion Policies:**
  1. **Restrict / Reject:** Disallow deletion of the referenced row if referencing rows exist.
  2. **Cascade:** Automatically delete all referencing rows.
  3. **Set NULL:** Set the foreign key attribute in referencing rows to `NULL`.

→ **Natural vs. Surrogate Keys:**

- **Natural Key:** Originates from the domain (e.g., email address). Can change over time and carries business meaning.
- **Surrogate Key:** System-generated identifier (e.g., UNI, CUID). Immutable and carries no business meaning.

6. Declarative Query Execution & SQL

SQL (Structured Query Language) is the standard declarative query language for relational databases (course uses PostgreSQL). Users specify *what* result they want, not *how* to compute it. The **Query Optimizer** evaluates table size, indexes, and hardware to choose the optimal execution strategy.

Basic Single-Table Query Structure & Duplicates

- **Clauses:** `SELECT` (attribute list) → `FROM` (relation) → `WHERE` (tuple predicate).
- **Duplicates:** SQL retains duplicate rows by default. Use `SELECT DISTINCT` to eliminate duplicate output rows.
- **Sorting:** Use `ORDER BY attr [ASC|DESC]` to explicitly order results.

NULL Values & Three-Valued Logic

- **NULL Concept:** Represents an unknown or not applicable value (not zero or empty string).
- **Three-Valued Logic:** Comparisons involving `NULL` evaluate to **UNKNOWN**.
- **WHERE Filtering:** `WHERE` keeps a tuple only when the predicate evaluates to **TRUE**.
- **Testing NULL:** Comparisons like `WHERE grade = NULL` or `WHERE grade <> 'A'` do not return `NULL` rows. Use `IS NULL` or `IS NOT NULL`.