**The E/R Model**

**E/R Basics: Entities & Relations**

1. High-Level E/R Model

2. Entities

3. Relations

**Database Design Process**

E/R is a Visual Syntax for a DB Design which is precise enough for technical points, abstracted enough for non-technical people.

Entities and Entity Sets

* Primitive unit of the E/R model

Entities are the *individual Objects*, which are members of entity sets

* **Example**: A specific person or product

Entity Sets are the classes or types of objects in our model

* **Example**: Person or Product

**Entities and Entity Sets**

An Entity Sets has attributes.

**Keys**

* A key is a minimal set of attributes that uniquely identifies an entity
* The E/R model forces us to designate a single *primary key*, we denote it by underling it in our E/R model

**The R in E/R: Relationships**

* A *relationship* is between two entities

**What Is a Relationships?**

* A Mathematical Definition:
  + Let A, B be sets and A x B (cross-product), we define a relationship to be a subset of A x B
* There can only be one relationship for every unique combination of entities, which means that the relationship is uniquely determined by the keys of its entities.
* Relationships may have attributes as well

**2. E/R Design Considerations**

1. Relationships Continued (Multiplicity, Multi-Way)
2. Design Considerations
3. Conversions to SQL
4. **Multiplicity of E/R Relationships**

One-to-One:

Many-to-One:

One-to-One:

Many-to-Many:

**3. From E/R Diagrams to Relational Schema**

**Lecture 2: Design Theory 1**

* Normal Forms and Functional Dependencies
  + Activity: Finding FDs
* Data Anomalies and Constraints
* Functional Dependencies
* FDs for Relational Schema Design

**Design Theory**

* Design Theory is about how to represent your data to avoid anomalies (not standard)

**Normal Forms**

**Functional Dependents**

Why Care?

* They allow us to decide whether a database design is correct

Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion, and update anomalies. I’s helpful to minimize the redundancy in relation.

**Normal Forms**

Good Database design is about ensuring the integrity of the data, we also want good performance.

What is a Normal Form?

Why is this important?

High level guidance on how to design a database

**Database Undercover**

Let’s learn how to build software that manages a database.

We will look at Disk-Oriented Architecture. The DBMS assumes that the primary location of the database is on non-volatile disk. The DBMS’s components manage the movement of data between non-volatile and volatile storage.

Non-Volatile means if we were to “unplug” the computer the memory/data wouldn’t be lost, while Volatile means that the memory/data would be lost.

**Buffer Manager and File Organization**

What is the Buffer Manager?

* We will consider a buffer located in main memory that operates over pages and files

Well, what is the buffer?

* Region of physical memory used to store temporary data
* In our Database, the buffer operates over pages and file

Basic Operations:

**Read(Page):** Reads a page from disk-> buffer if not already in buffer

**Flush(Page):** Evict Page from buffer & write to disk

The Buffer Manager handles supporting operations for the buffer that primarily handles and executes the “replacement policy”

* I.E finds a page in buffer to flush/release if buffer is full and a new page needs to be read in

DBMS typically implement their own buffer management routines.

**Simplified Filesystem Model:**

For use a page is a **Fixed-Sized Array** of memory.

A File is a variable-length list of pages.

