CSE 510
Web Data Engineering

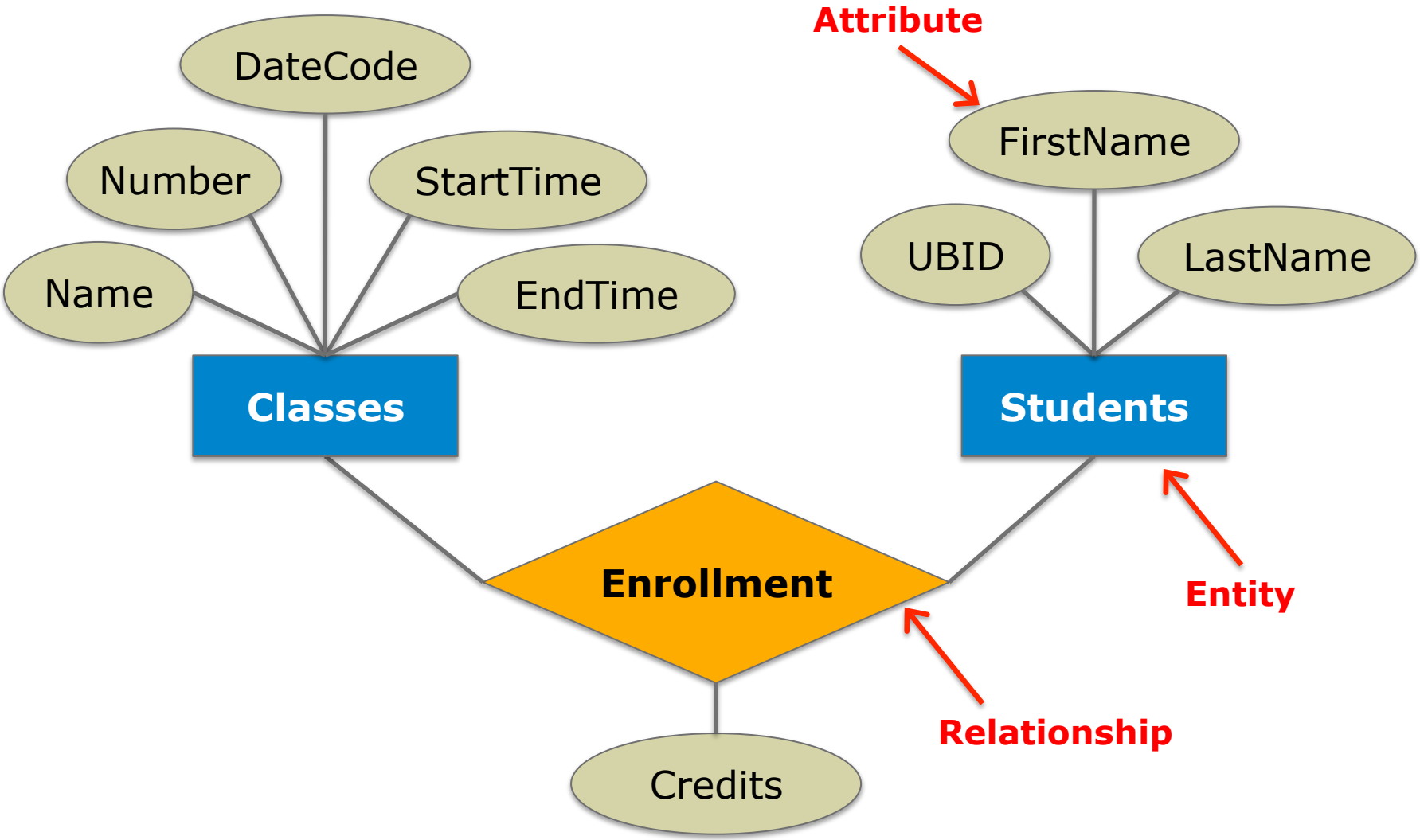
Database Design

cse@buffalo

How to Design a Database and Avoid Bad Decisions

- With experience…
- Learn in CSE462 normalization rules of database design
- Think **entities and relationships** – translate to relations

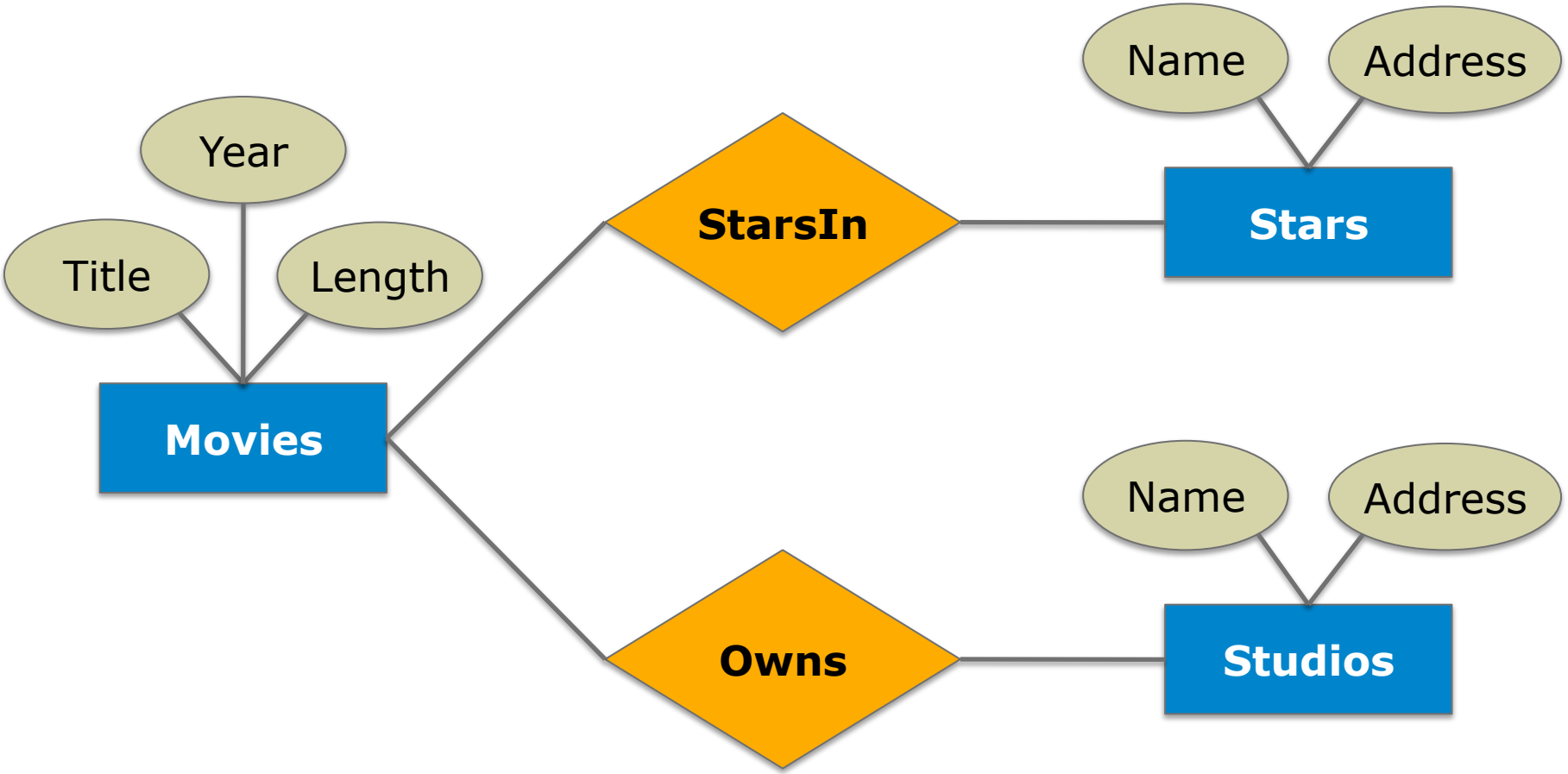
E/R-Based Design



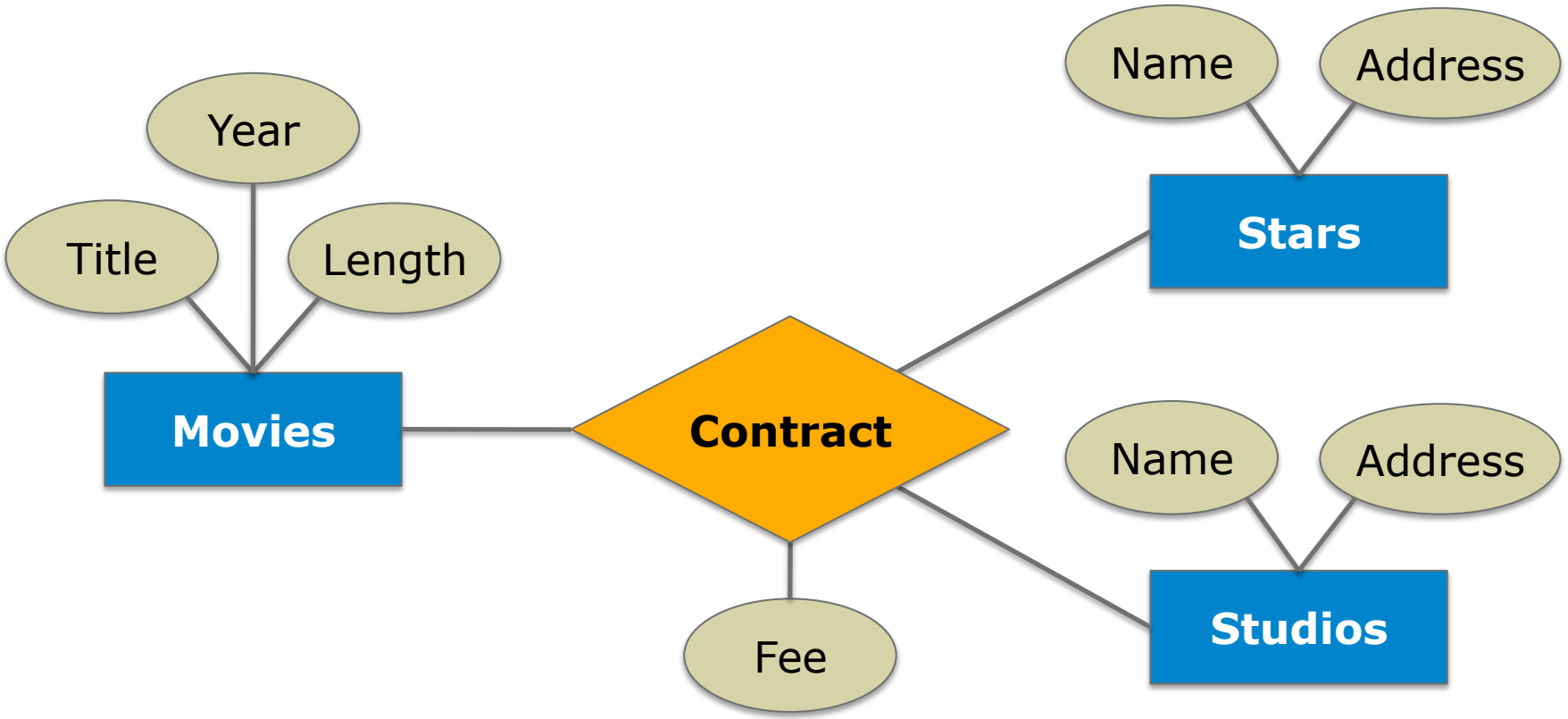
E/R → Relational Schema: Basic Translation

- For every entity, create corresponding table
  - Include an ID attribute even if not in E/R
- For every relationship, create table
  - For each referenced entity E_i include foreign key attribute referencing ID of E_i

Example

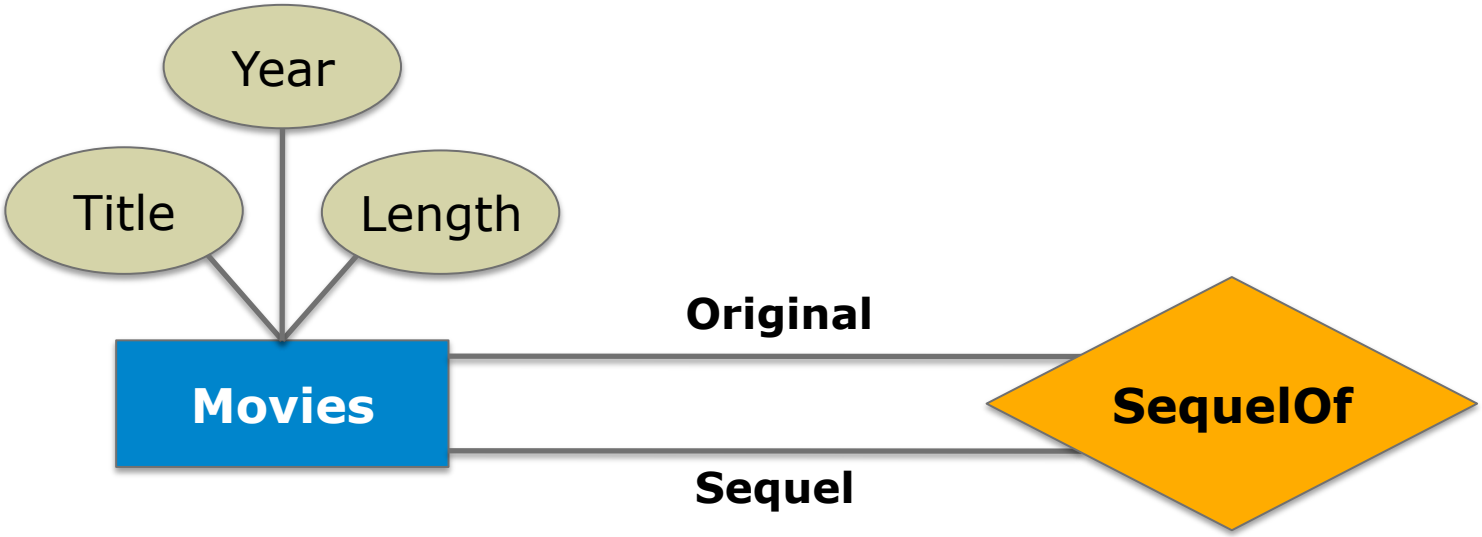

Name
Address
Year
StarsIn
Stars
Title
Length
Movies
Name
Address
Owns
Studios

3-Way Relationship

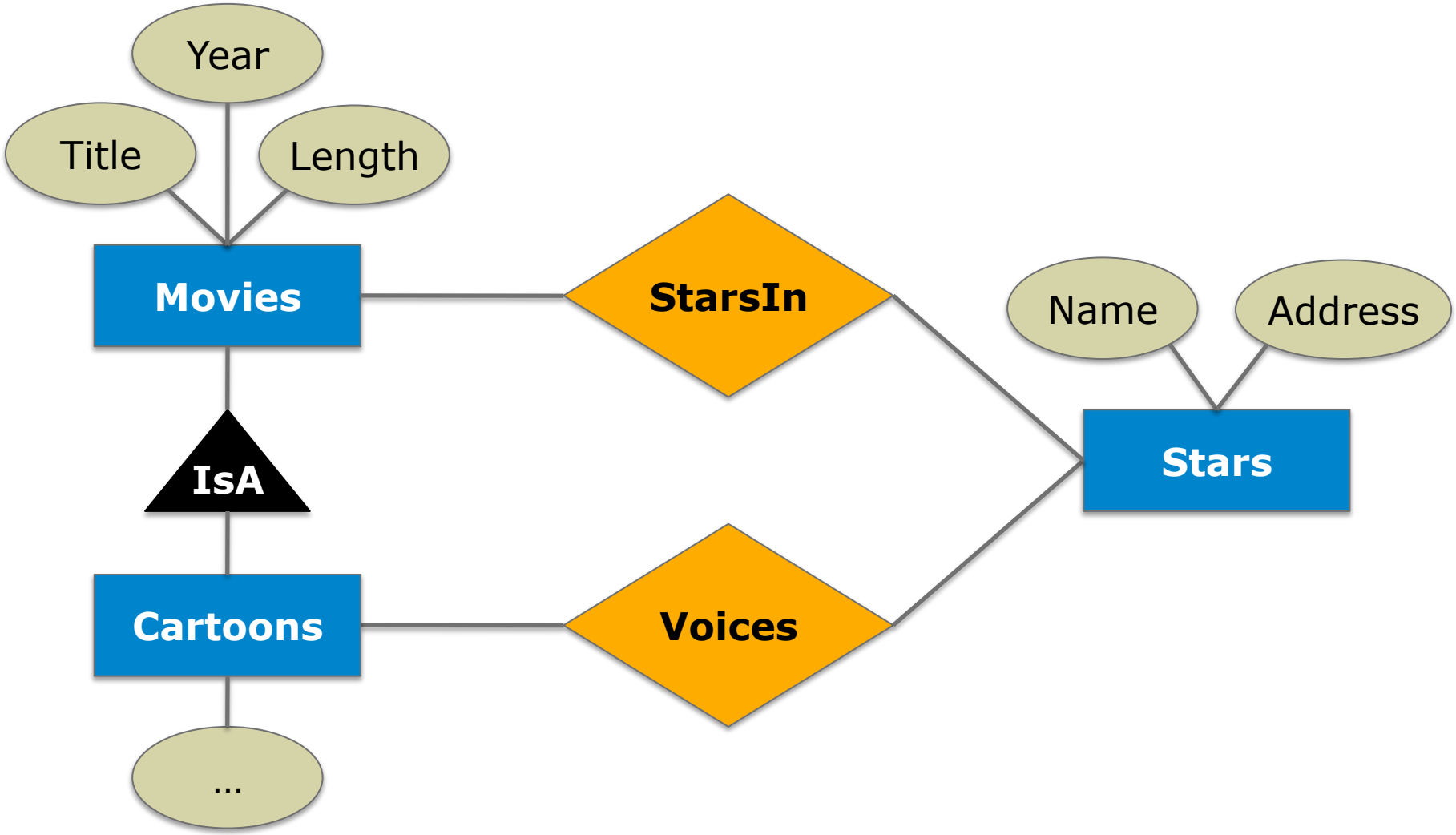


- A studio has contracted with a particular star to act in a particular movie

Relationships with Roles



"Subclassing"



Transaction Management

- **Transaction:** Collection of actions that maintain the consistency of the database if ran to completion & isolated
- **Goal:** Guarantee integrity and consistency of data despite
  - Concurrency
  - Failures
- Concurrency Control
- Recovery

Concurrency & Failure Problems

- Consider the "John & Mary" checking & savings account
  - C: checking account balance
  - S: savings account balance
- Check-to-Savings transfer transaction moves $X from C to S
  - If it runs in the system alone and to completion, the total sum of C and S stays the same

```
C2S(X=100)
Read(C)
C:=C-100
Write(C)
Read(S)
S:=S+100
Write(S)
```

Failure Problem & Recovery Module's Goal

```
C2S(X=100)
Read(C)
C:=C-100
Write(C)

CPU Halts
Read(S)
S:=S+100
Write(S)
```

- Database is in inconsistent state after machine restarts
- It is not the developer's problem to account for crashes
- Recovery module guarantees that all or none of a transaction happens and its effects become "durable"

Concurrency Problem & Concurrency Control Module's Goals

Serial Schedule

```
Read(C)
C:=C+100
Write(C)
Read(S)
S:=S-100
Write(S)
                Read(C)
                C:=C+50
                Write(C)
                Read(S)
                S:=S-50
                Write(S)
```

- If multiple transactions run in sequence, the resulting database is consistent
- Serial schedules
  - De facto correct

Concurrency Problem & Concurrency Control Module's Goals

Good Schedule with Concurrency

```
Read(C)
C:=C+100
Write(C)
                Read(C)
                C:=C+50
                Write(C)
Read(S)
S:=S-100
Write(S)
                Read(S)
                S:=S-50
                Write(S)
```

- Databases allow transactions to run in parallel

Concurrency Problem & Concurrency Control Module's Goals

Bad Schedule with Concurrency

```
Read(C)
C:=C+100
                Read(C)
Write(C)
                C:=C+50
                Write(C)
                Read(S)
                S:=S-50
                Write(S)
Read(S)
S:=S-100
Write(S)
```

- "Bad" interleaved schedules may leave database in inconsistent state
- Developer should not have to account for parallelism
- Concurrency control module guarantees **serializability**
  - only schedules equivalent to serial ones happen