

## Roadmap for OO Design

- RDBMS principles
- Implementing OO models in RDBMS
  - Normalization and good models
- More OO design
  - polymorphism, enhancing inheritance
- Architecture
  - Alternate Data management
  - External control
  - Reuse

## What is a Relational Database?

Database = collection of tables  
 Table = collections of rows  
 Row = set of attributes(fields)

**Students**

Student ID	Student Name	Faculty ID
123	French	KS
256	Smith	WM
301	Jones	GS

attributes →

## Domains and Keys

Each attribute defines a **Domain**: Set of values for attribute

Student ID={x | x is a valid student ID}

Student Name={x | x is a student name}

Faculty ID={x | x is faculty ID initials}

A Key is a set of attributes that identifies a row.

## Key Nomenclature

**Candidate Key**: Uniquely identifies a row.

**Primary Key**: a Candidate Key used for main access

**Foreign Key**: An attribute from one table that is a Primary Key in another table.

## Keys Example

Prof ID	Name	Dept	Office
M21	McUmbert	CSE	3100
P01	Potter	EEC	3200
S56	Stirewalt	CSE	12-321
CAA	Crowe	CEP	45-004

Primary key

Dept	Chair	Fax#
CSE	Stockman	355-1111
EEC	Rover	353-1212
CEP	Leheay	432-1234
DUV	Leheay	355-4444

Foreign key

Note: referential integrity

## Why is it called Relational?

Recall:

Given sets  $A, B, C, \dots$   $A \times B \times C = (a, b, c)$  such that  $a \in A, b \in B, c \in C$ ,  $A \times B \times C$  is called a Cross Product.

The sets  $A, B, C$  are the **domains** (attribute values)

A relation  $R \subset A \times B$  is a function if

## Operations on Tables

All the RDBMS operators act on tables to produce tables.

**Project** operator = all rows, but selected columns

**Select** operator = select rows based on criteria (such as key match)

**Join** A, B over column x from A, y from B = (two steps):

1. Form  $C = A \times B$

2. Keep only rows where  $x=y$

## Join Example

Table A			Table B		
Student ID	Name	Faculty ID	Faculty ID	Section	Location
124	French	A	A	101	EB 201
256	Smith	B	B	201	BH 100
301	Jones	C	C	001	EB 304

A.Student ID	A.Name	A.Faculty	B.Faculty	B.Section	B.Location
124	French	A	A	101	EB 201
256	Smith	B	B	201	BH 100
301	Jones	C	C	001	EB 304

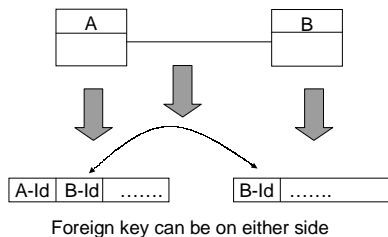
## Support for Existence-Based Identity

- RDBMS generates unique ID (usually number)
- MS-Access uses type **counter** (really a long value)
- Oracle uses type **sequence**
- Other possibilities
  - Date/Time (if fine grained)
  - pointer connected with DB (Ingres)

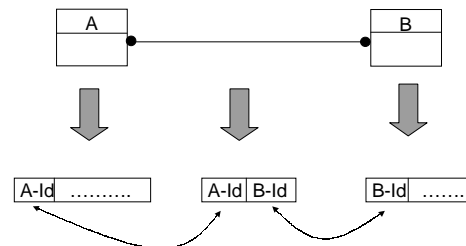
## Implementing a Class Model

- Objects have identity
  - Value based (keys are domain values)
  - Existence based (keys are made-up Ids)
- New Domains
  - Identifier: contains existence Ids
  - Enumeration: discrete set of values
- Class  $\Rightarrow$  table
- Assoc & Generalization  $\Rightarrow$  table or relation

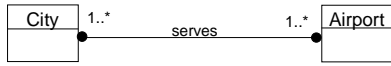
## Implementing 1-1



## Implementing Many to Many

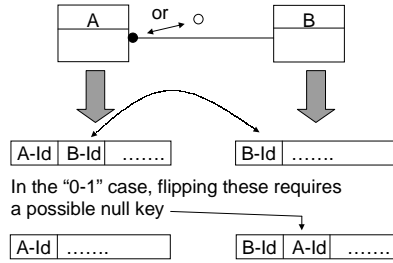


## Example with Foreign Keys

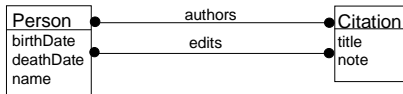


City Table		Serves Table		Airport Table	
cityID	cityName	cityID	airportCode	airportCode	airportName
01005	Houston	01005	IAH	IAH	Intercontinental
01006	Atlanta	01005	HOU	HOU	Hobby
01007	Albany	01008	TEW	ATL	Hartsfield
01009	Lansing	01009	LAN	LAN	Lansing
		01006	ATL		

## Implementing 1 to Many

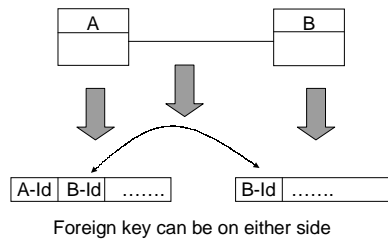


## Exercise

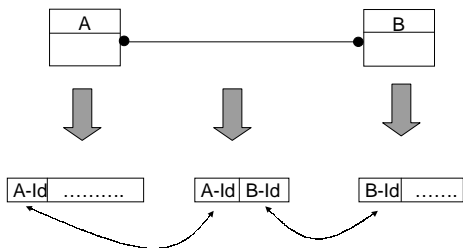


1. Use existence-based identity to design tables for classes Person and Citation.
2. Design tables for associations *authors* and *edits*.

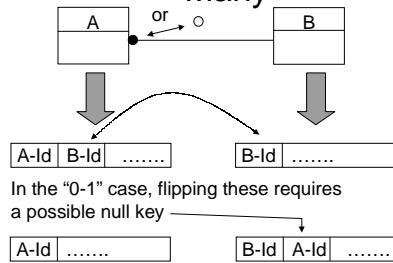
## Review: Implementing 1-1



## Review: Implementing Many to Many



## Review: Implementing 1 to Many



## Normal Forms and OO Implementations (1)

Schedule

Course	Section	Desc	Time	Credits
CSE 260	1	Math	8:00 MTF	3
CSE 470	1	Software	11:30MWF	4
CSE 260	2	Math	4:10MWF	3
CSE 101	3	Intro	9:10 MTF	3
CSE 231	2	C++	10:20 TTh	3

Primary key

If the description of CSE260 changes, have to change it in multiple places. Called **modification anomaly**.

## Normal Forms

- “Normal Form” old RDBMS concept
- Concerned with redundancies and dependencies between attributes.
- With Object Models we can (almost) ignore normalization
- Requires good model and good implementation.

## Normal Forms and OO Implementations (2)

Schedule

Course	Section	Desc	Time	Credits
CSE 260	1	Math	8:00 MTF	3
CSE 470	1	Software	11:30MWF	4
CSE 260	2	Math	4:10MWF	3
CSE 101	3	Intro	9:10 MTF	3
CSE 231	2	C++	10:20 TTh	3

If I want to add CSE870, and have a description and credits, I still need a section and time before it can be added. This is an **insertion anomaly**.

## Normal Forms and OO Implementations (3)

Schedule

Course	Section	Desc	Time	Credits
CSE 260	1	Math	8:00 MTF	3
CSE 470	1	Software	11:30MWF	4
CSE 260	2	Math	4:10MWF	3
CSE 101	3	Intro	9:10 MTF	3
CSE 231	2	C++	10:20 TTh	3

If no sections of CSE101 are offered, and I delete it, I lose the description and credits. This is a **deletion anomaly**.

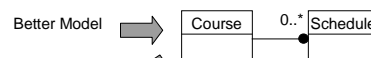
## Normal Forms & Correct Models

Schedule

Course	Section	Desc	Time	Credits
CSE 260	1	Math	8:00 MTF	3
CSE 470	1	Software	11:30MWF	4
CSE 260	2	Math	4:10MWF	3
CSE 101	3	Intro	9:10 MTF	3
CSE 231	2	C++	10:20 TTh	3

This table has problems because it has partial dependencies, and is not in **second normal form**. It's also the implementation of an incorrect object model. Description and Credits only depend on Course.

## Second Normal Form



Course			Schedule		
Course	Desc	Credits	Course	Section	Time
CSE 260	Math	3	CSE 260	1	8:00 MTF
CSE 470	Software	4	CSE 470	1	11:30 MWF
CSE 101	Intro	3	CSE 101	3	9:10 MTF
CSE 231	C++	3	CSE 231	2	10:20 TTh

**Second normal form** means there are no partial dependencies.

## Third Normal Form (I)

Prof						
Prof ID	Name	Dept	Office	Chair	FAX	
B1	Brown	CSE	263	Stockman	555-7441	
J2	Jones	EEC	241	Fui	555-7318	
M3	Morris	CSE	127	Stockman	555-7441	
W4	West	PSY	257	Freud	505-0012	

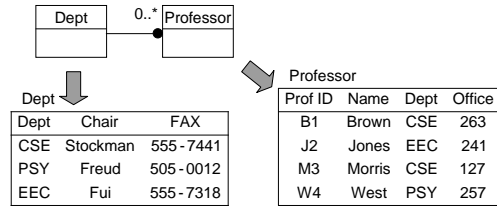
This "professor" class is modeled incorrectly. The table is in second normal form, but has anomalies:

**Modification:** Changing dept chair requires multiple modifications

**Insertion:** A new Dept requires at least one prof even if we have a chair and FAX.

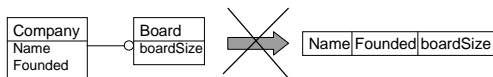
Why? Chair and FAX depend on only Dept. Called **Transitive Dependency**

## Third Normal Form (II)

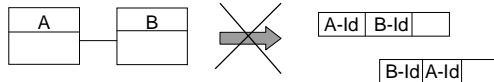


Now there are no partial or transitive dependencies.

## Things NOT To Do



Don't combine optional classes

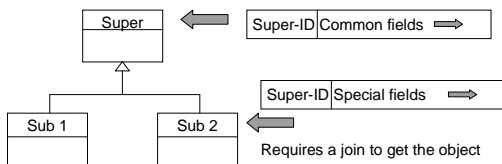


Don't double up on foreign keys

## Conclusions So Far....

- Classes are tables
- Associations are relations or tables and relations
- Proper class model produces normalized forms
- Normalized forms eliminate anomalies
- Concentrate on class model -- proper classes and multiplicities and good relational model results.

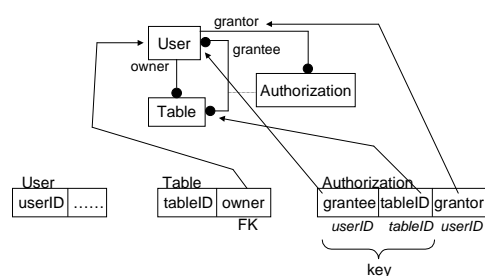
## Generalization



Alternatives: Push subclass attributes up and use a "type" field

Push superclass attributes down and replicate tables

## Association Classes



## Link Attributes

Like any other association table, but add the extra attributes.

