Database Tables

IOE 373 Lecture 03

Topics

I. Table Keys

II. Table Design Guidelines

III. Normalization

I. Primary Keys

- Definition: A primary key is a column or set of columns that uniquely identifies each row in a table.
- A key is a way to identify a particular record in a table. The primary key is a field, or collection of fields, that allows for quick access to a record.

Candidate Keys

- For some entities, there may be more than one field or collection of fields that can serve as the primary key.
- These are called candidate keys.

Candidate Keys

- All candidate keys can open the lock;
- That is, they can uniquely identify a record in a table.
- Consider a table containing U of M students.
- What are some possible candidate keys?

Compound Keys

- The primary key doesn’t have to be a single column. In many cases, the combination of two or more fields can serve as the primary key. For example, consider a table of all of the courses offered at the University.
- The course number would not be sufficient to identify a single course; there are 101’s in many departments, and probably plenty of 373’s as well.
- Department won’t work either; there are many courses in IOE, as well as Psychology, German, and Economics.
- But when you combine department and course number, you have uniquely identified a course.
- Therefore, those two fields together can serve as the primary key for the Course table.
- ***Definition--Compound Key: A primary key consisting of two or more fields.***
- ***Definition—Simple Key: A primary key consisting of one field.***

Choosing a Key

- Picking/assigning a primary key can be tricky, and is a matter of some controversy.
- Some books recommend that you always create a surrogate or artificial key; by default, Access does this for you.
- General approach:
  1. if the table already has a natural primary key use it;
  2. if not, then use a combination of fields that guarantees uniqueness;
  3. only if you can't find a natural simple or compound key should you resort to creating a surrogate key.

“Natural” Keys

- Many entities already have widely accepted and enforced keys that you can use in your database—these are termed “natural” keys.
- ***Definition: A “natural key” is a pre-existing or ready-made field which can serve as the primary key for a table.***
- For employees, social security number is widely used. You can have two Tim Joneses working for you, but only one 123-45-6789.
- For vehicles, you can use Vehicle Identification Number (VIN), a unique 17-character code that every vehicle manufactured is required to have.
- Most “natural” keys aren’t really natural; they are simply artificial or “surrogate” keys that someone else created and that are now widely recognized.

Choosing a Primary Key

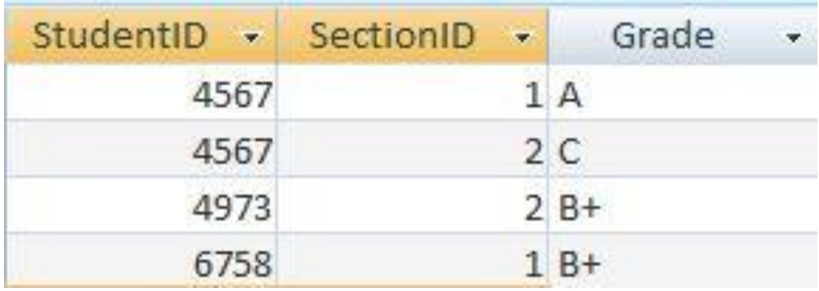
- If there is only one candidate, choose it.
- Choose the candidate **least likely to have its value changed**.
  - Changing primary key values once we store the data in tables is a complicated matter because the primary key can appear as a foreign key in many other tables.
- Choose the simplest candidate. The one that is composed of the fewest number of attributes is considered the simplest.
- Choose the shortest candidate. This is purely an efficiency consideration. However, when a primary key can appear in many tables as a foreign key, it is often worth it to save some space with each one.

Surrogate Keys

- Occasionally, you will come across a table that has NO candidate keys.
- In this case, you must resort to a surrogate key ***Definition: A "surrogate key" is a field (usually numeric) added to a table as the primary key when no natural keys are available.***
- When you have to resort to a surrogate key, don't try to put any meaning into it.
- The values in surrogate key fields frequently become important identification numbers: Social Security Numbers, Student IDs, VINs, SKUs (in stores), etc. They are frequently associated with some sort of ID card or tag.

Formatting Conventions

- Tables should generally be named as plural nouns, with the first letter capitalized: Customers, Orders, Products, etc.
- Table names should not have any spaces or punctuation in them. The following are bad table names: Bob’s Customers, Back Orders, hours/day. (Details later.)
- When designing a table, the primary key field or fields should be at the far left of the table.
- It is common to see the primary key field(s) highlighted in some way—underlined, bold, with an asterisk, etc.
- For example, we can highlight the primary key in yellow, like this:



StudentID	SectionID	Grade
4567	1	A
4567	2	C
4973	2	B+
6758	1	B+

Rules for Table and Field Names

- No Spaces! Access will allow you to name a table “My Company’s Employees”, but this causes many problems down the road, especially when interfacing with VB.
- “Employees” would be a much better name
- The whole database probably relates to “My Company”, so there’s no need to include that in the table name. Even if there were—NO SPACES!!!!!

Rules for Table and Field Names

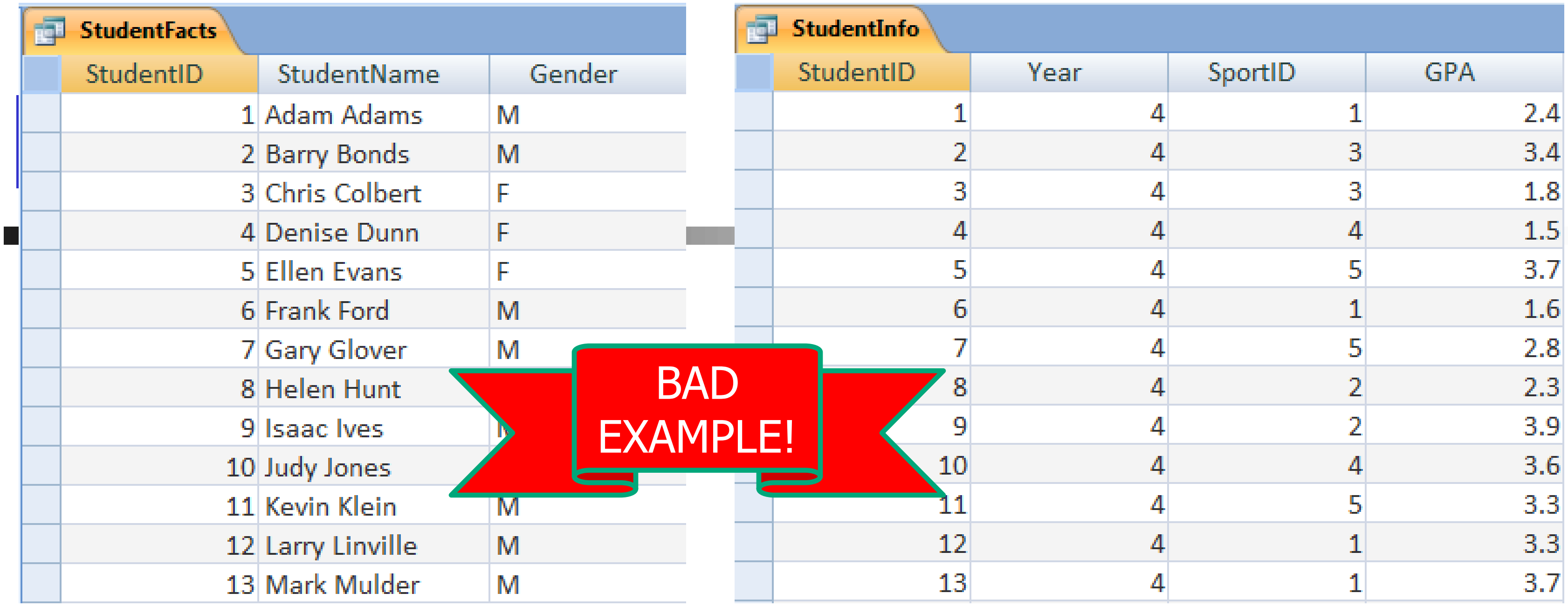
- No punctuation: Most DBMS’s (except for Access) won’t allow any punctuation in the name of a field or table except the underscore (_).
- I’m not a fan of underscores, either—they tend to be obscured by hyperlinks and such.
- For this class, if your table or field needs a multi-word name, use **InteriorCapitals**.

Foreign Keys

- ***Definition: A foreign key is a field (or fields) in a table that is not the primary key in that table, but IS the primary key in another table.***
- Foreign keys are used for creating relationships (linking tables together)

II. Table Design

- Databases for real businesses tend to have a lot of tables, but not always the right number.
- Normalization generally results in more tables.
- However, beginning database designers frequently create too many tables in ways that have nothing to do with normalization. The most common of these are:
  - Using two tables in a one-to-one relationship.
  - Making separate tables based on an attribute.

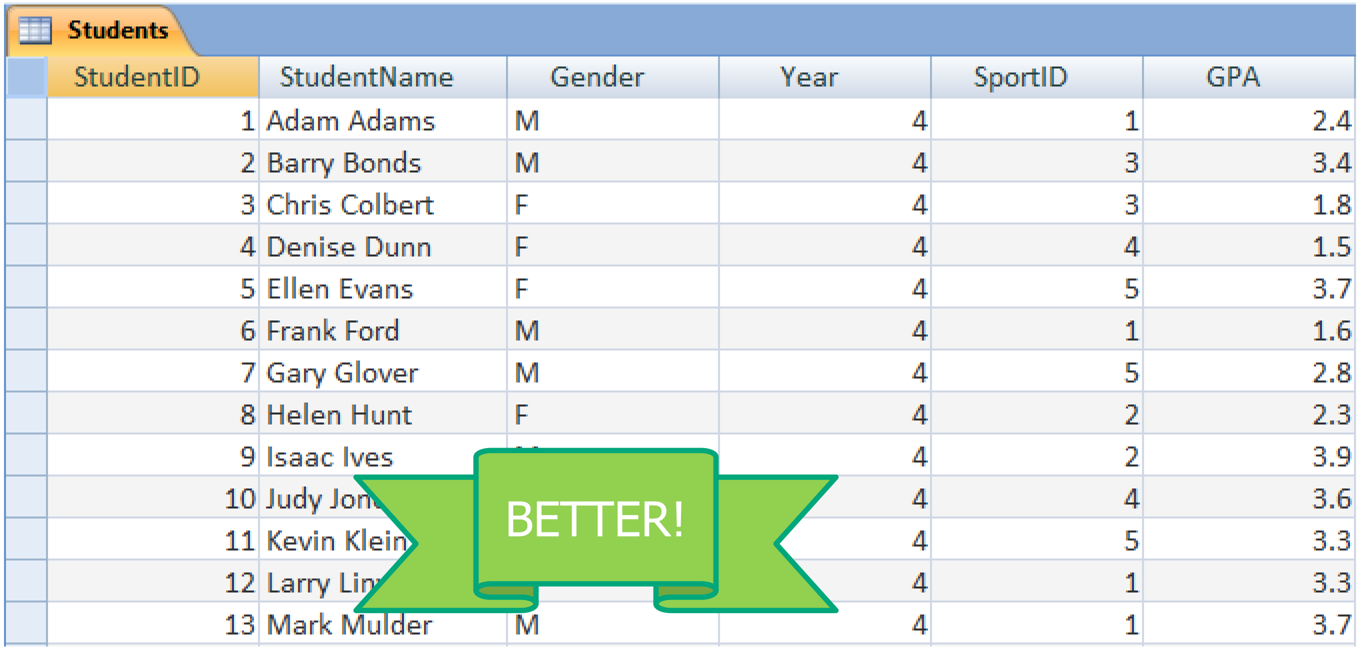


StudentFacts

StudentID	StudentName	Gender
1	Adam Adams	M
2	Barry Bonds	M
3	Chris Colbert	F
4	Denise Dunn	F
5	Ellen Evans	F
6	Frank Ford	M
7	Gary Glover	M
8	Helen Hunt	[illegible]
9	Isaac Ives	M
10	Judy Jones	[illegible]
11	Kevin Klein	M
12	Larry Linville	M
13	Mark Mulder	M

StudentInfo

StudentID	Year	SportID	GPA
1	4	1	2.4
2	4	3	3.4
3	4	3	1.8
4	4	4	1.5
5	4	5	3.7
6	4	1	1.6
7	4	5	2.8
8	4	2	2.3
9	4	2	3.9
10	4	4	3.6
11	4	5	3.3
12	4	1	3.3
13	4	1	3.7



Students

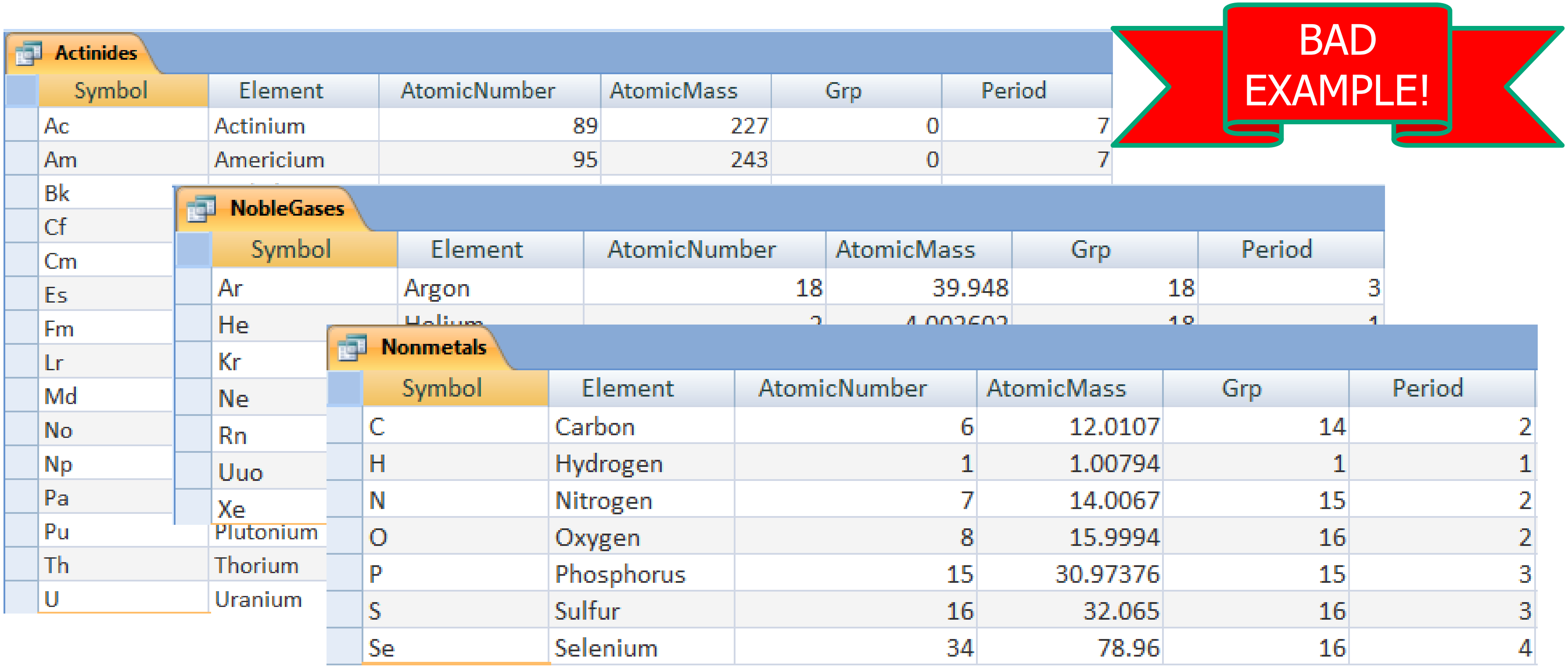
StudentID	StudentName	Gender	Year	SportID	GPA
1	Adam Adams	M	4	1	2.4
2	Barry Bonds	M	4	3	3.4
3	Chris Colbert	F	4	3	1.8
4	Denise Dunn	F	4	4	1.5
5	Ellen Evans	F	4	5	3.7
6	Frank Ford	M	4	1	1.6
7	Gary Glover	M	4	5	2.8
8	Helen Hunt	F	4	2	2.3
9	Isaac Ives	[illegible]	4	2	3.9
10	Judy Jon	[illegible]	4	4	3.6
11	Kevin Klein	[illegible]	4	5	3.3
12	Larry Lin	[illegible]	4	1	3.3
13	Mark Mulder	M	4	1	3.7

Separating Tables by an Attribute

- The most common type of error is creating multiple tables for a single entity, separating the records based on the value of a single attribute.
- This results in a database with a lot of tables which is slow and difficult to query.

Too Many Tables

- It is not uncommon for beginning database designers to think that different tables are used to represent different categories.
- Here is a design for a database meant to hold the chemical elements.



BAD EXAMPLE!

Actinides

Symbol	Element	AtomicNumber	AtomicMass	Grp	Period
Ac	Actinium	89	227	0	7
Am	Americium	95	243	0	7
Bk					
Cf					
Cm					
Es					
Fm					
Lr					
Md					
No					
Np					
Pa					
Pu	Plutonium				
Th	Thorium				
U	Uranium				

NobleGases

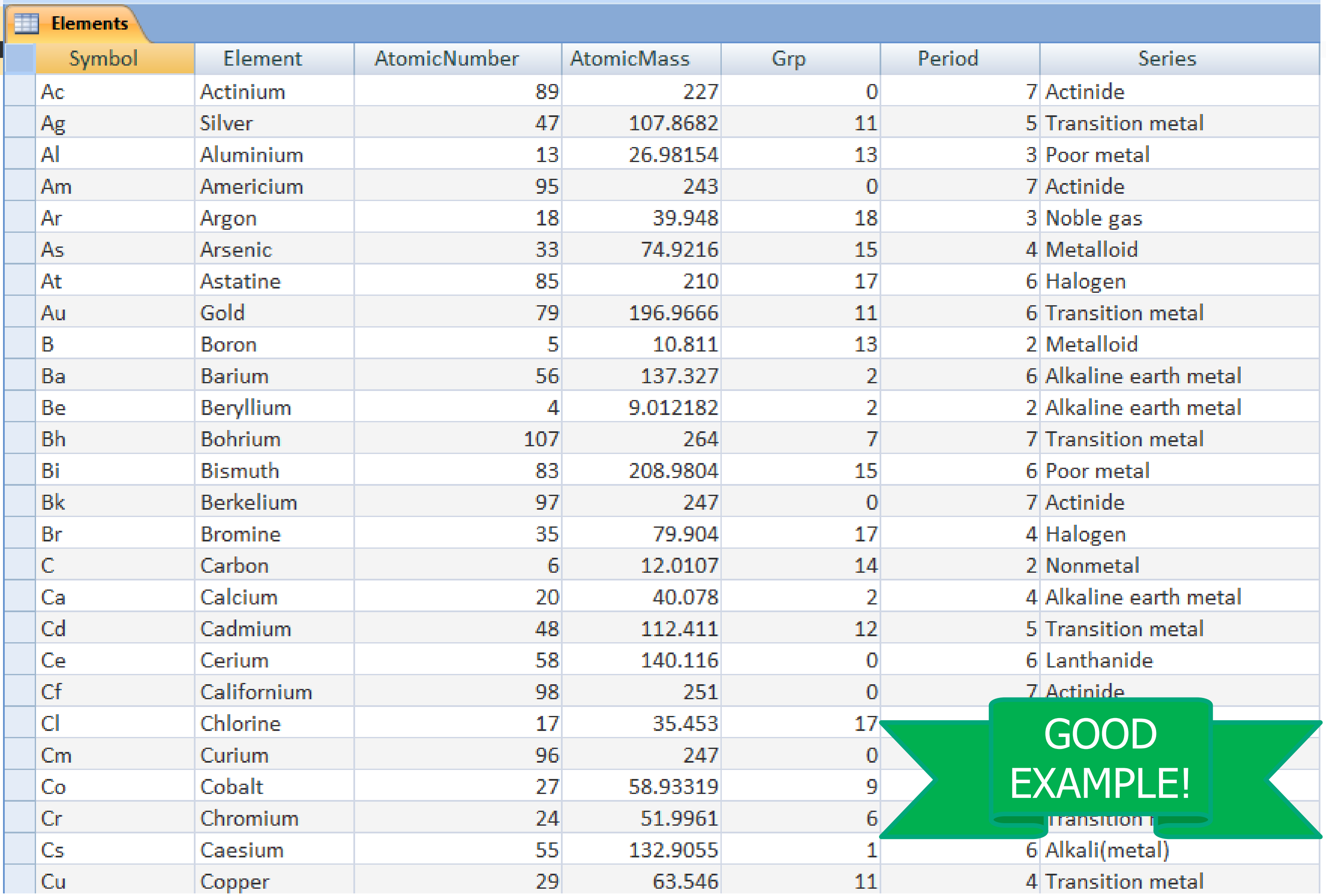
Symbol	Element	AtomicNumber	AtomicMass	Grp	Period
Ar	Argon	18	39.948	18	3
He	Helium	2	4.002602	18	1
Kr					
Ne					
Rn					
Uuo					
Xe					

Nonmetals

Symbol	Element	AtomicNumber	AtomicMass	Grp	Period
C	Carbon	6	12.0107	14	2
H	Hydrogen	1	1.00794	1	1
N	Nitrogen	7	14.0067	15	2
O	Oxygen	8	15.9994	16	2
P	Phosphorus	15	30.97376	15	3
S	Sulfur	16	32.065	16	3
Se	Selenium	34	78.96	16	4

- As you can see, each table has exactly the same fields.
- The only thing separating the tables is the “Series” of the elements—Actinides, NobleGases, Nonmetals, etc.
- By recognizing that Series is really just another attribute of elements, all of these tables can be combined into one table containing all elements.

- Adding a “Series” column allows all of the elements to be stored in a single table.



Elements

Symbol	Element	AtomicNumber	AtomicMass	Grp	Period	Series
Ac	Actinium	89	227	0	7	Actinide
Ag	Silver	47	107.8682	11	5	Transition metal
Al	Aluminium	13	26.98154	13	3	Poor metal
Am	Americium	95	243	0	7	Actinide
Ar	Argon	18	39.948	18	3	Noble gas
As	Arsenic	33	74.9216	15	4	Metalloid
At	Astatine	85	210	17	6	Halogen
Au	Gold	79	196.9666	11	6	Transition metal
B	Boron	5	10.811	13	2	Metalloid
Ba	Barium	56	137.327	2	6	Alkaline earth metal
Be	Beryllium	4	9.012182	2	2	Alkaline earth metal
Bh	Bohrium	107	264	7	7	Transition metal
Bi	Bismuth	83	208.9804	15	6	Poor metal
Bk	Berkelium	97	247	0	7	Actinide
Br	Bromine	35	79.904	17	4	Halogen
C	Carbon	6	12.0107	14	2	Nonmetal
Ca	Calcium	20	40.078	2	4	Alkaline earth metal
Cd	Cadmium	48	112.411	12	5	Transition metal
Ce	Cerium	58	140.116	0	6	Lanthanide
Cf	Californium	98	251	0	7	Actinide
Cl	Chlorine	17	35.453	17		
Cm	Curium	96	247	0		
Co	Cobalt	27	58.93319	9		
Cr	Chromium	24	51.9961	6		Transition
Cs	Caesium	55	132.9055	1	6	Alkali(metal)
Cu	Copper	29	63.546	11	4	Transition metal

GOOD EXAMPLE!

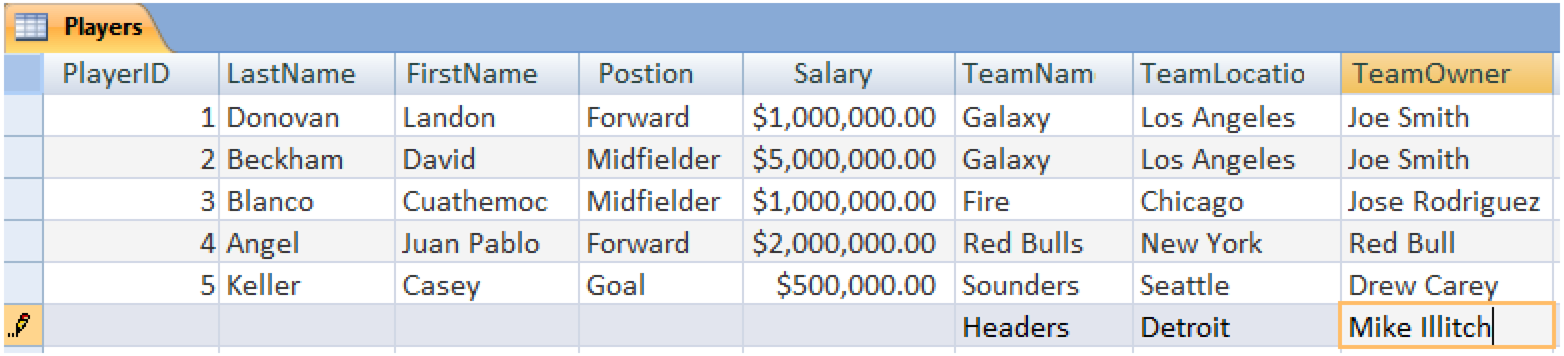
Same Fields, Same Table

- If you have two tables that have exactly the same fields, they almost certainly represent the same entity. Therefore,
- The tables should be combined, adding a field to hold the attribute that you used to separate them.

III. Normalization: Three Anomalies

- The three anomalies are problems caused by trying to put data for more than one entity into a single table.
- The anomalies are:
  - Insert
  - Update
  - Delete

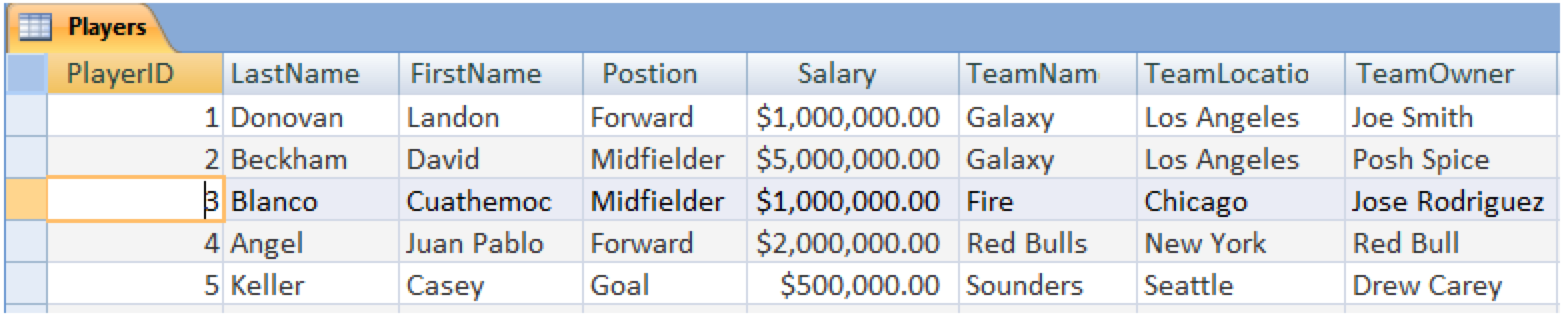
Insert Anomaly



PlayerID	LastName	FirstName	Postion	Salary	TeamNam	TeamLocatio	TeamOwner
1	Donovan	Landon	Forward	$1,000,000.00	Galaxy	Los Angeles	Joe Smith
2	Beckham	David	Midfielder	$5,000,000.00	Galaxy	Los Angeles	Joe Smith
3	Blanco	Cuathemoc	Midfielder	$1,000,000.00	Fire	Chicago	Jose Rodriguez
4	Angel	Juan Pablo	Forward	$2,000,000.00	Red Bulls	New York	Red Bull
5	Keller	Casey	Goal	$500,000.00	Sounders	Seattle	Drew Carey
					Headers	Detroit	Mike Illitch

- Above is a partial table of Major League Soccer players.
- To save time, the database manager has put the team information into the players table.
- When I try to add a Detroit expansion team, Access won’t let me because I don’t yet have any players associated with the team.
- To add a record to this table, I must have a PlayerID to enter as the primary key.
- This is the insert anomaly.

Update Anomaly

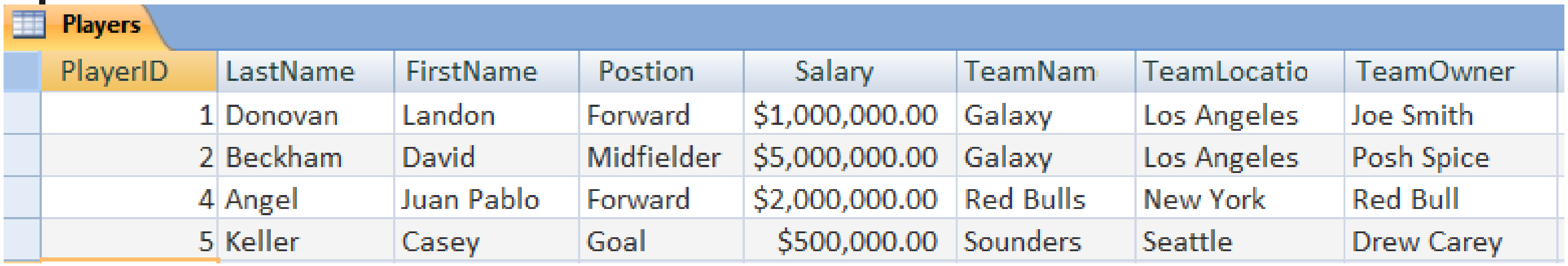


Players

PlayerID	LastName	FirstName	Postion	Salary	TeamNam	TeamLocatio	TeamOwner
1	Donovan	Landon	Forward	$1,000,000.00	Galaxy	Los Angeles	Joe Smith
2	Beckham	David	Midfielder	$5,000,000.00	Galaxy	Los Angeles	Posh Spice
3	Blanco	Cuathemoc	Midfielder	$1,000,000.00	Fire	Chicago	Jose Rodriguez
4	Angel	Juan Pablo	Forward	$2,000,000.00	Red Bulls	New York	Red Bull
5	Keller	Casey	Goal	$500,000.00	Sounders	Seattle	Drew Carey

- David Beckham’s wife decides to buy his team.
- You duly record this in his player record.
- But now you have Landon Donovan and David Beckham playing for the same team, but different owners!
- Your data is now inconsistent.
- This is the update anomaly.

Delete Anomaly



Players

PlayerID	LastName	FirstName	Postion	Salary	TeamNam	TeamLocatio	TeamOwner
1	Donovan	Landon	Forward	$1,000,000.00	Galaxy	Los Angeles	Joe Smith
2	Beckham	David	Midfielder	$5,000,000.00	Galaxy	Los Angeles	Posh Spice
4	Angel	Juan Pablo	Forward	$2,000,000.00	Red Bulls	New York	Red Bull
5	Keller	Casey	Goal	$500,000.00	Sounders	Seattle	Drew Carey

- Cuauhtémoc Blanco suffers a career-ending injury playing for Mexico in the World Cup. ¡Qué lástima!
- You remove him from your MLS roster.
- Oh-oh! You’ve removed all your information about the Chicago Fire team as well!
- This is the delete anomaly.

One Entity Per Table

- **All three anomalies result from the same cause: Putting attributes of multiple entities into a single table**.
- Codd and others recognized this, and codified (Coddified?) the principles of table and database design that prevent this from happening.
- Implementing these principles is called “normalization,” and the goal of normalization is, in most cases, Third Normal Form.
- I will abbreviate Third Normal Form as “3NF”.

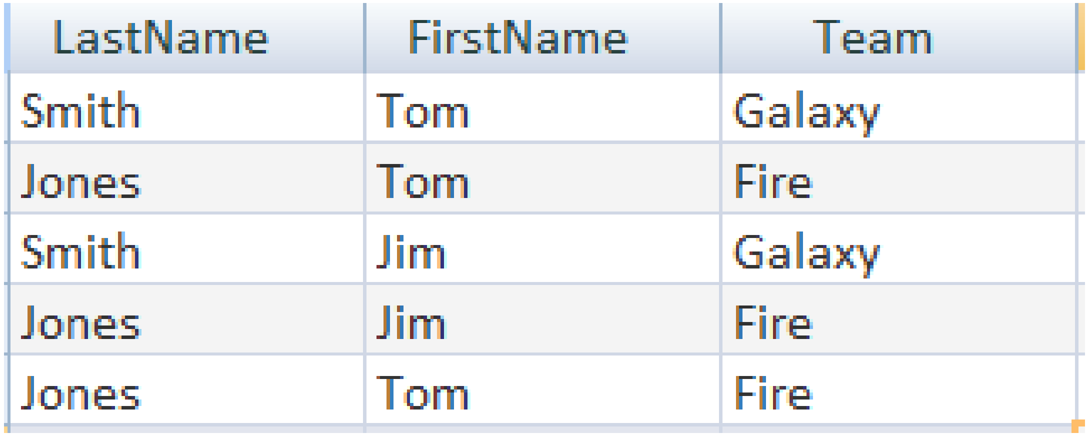
Normalization

- Most of the database books refer to normalization as a process—you start from some horribly messed-up database, put it through your normalizer, and out comes a proper database in 3NF.
- In general, I don’t think it works that way.
- If you are designing a new database from scratch, you can stick to the one-entity-per-table rule (and a few other rules) and create a properly normalized database the first time.

First Normal Form (1NF)

- First Normal Form is poorly defined in most books, and the books generally don’t agree with each other.
- It is easiest to define negatively.
- The next few slides will describe the things that keep a table from being in 1NF.

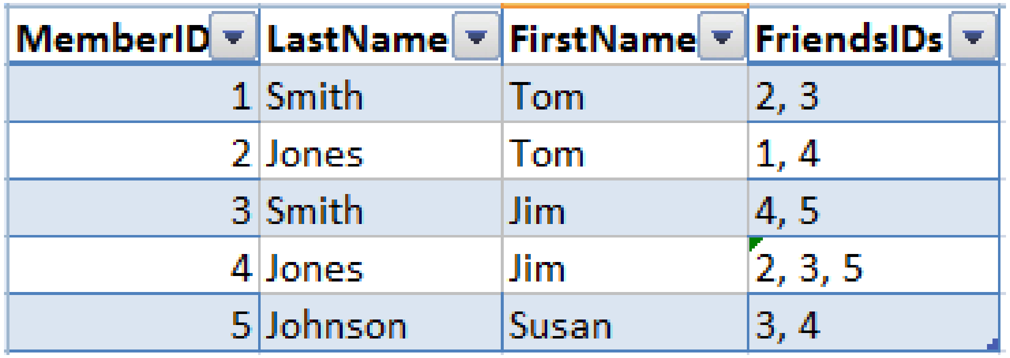
A table is ***NOT*** in 1NF if it doesn’t have a **primary key**



LastName	FirstName	Team
Smith	Tom	Galaxy
Jones	Tom	Fire
Smith	Jim	Galaxy
Jones	Jim	Fire
Jones	Tom	Fire

- A table without a primary key has no way to keep out duplicate entries.

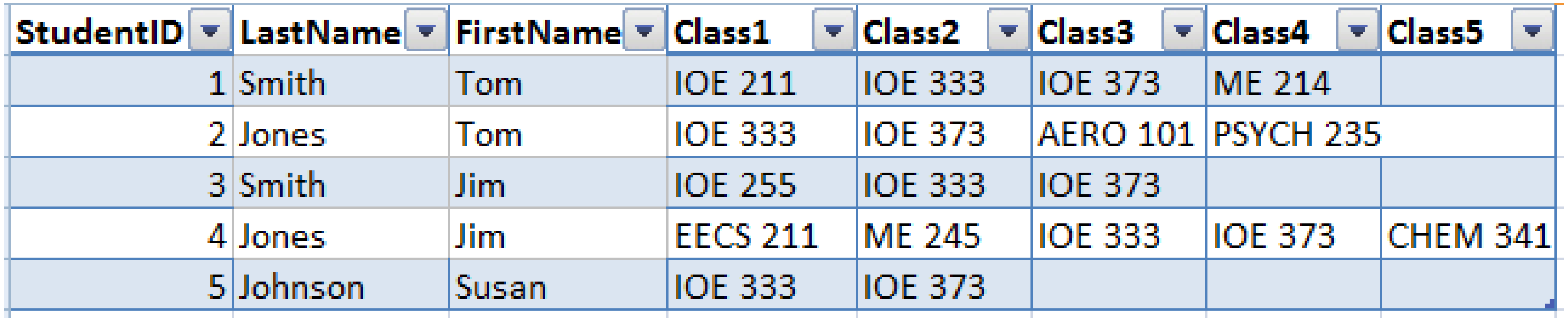
A table is ***NOT*** in 1NF if cells contain multiple data items



MemberID	LastName	FirstName	FriendsIDs
1	Smith	Tom	2, 3
2	Jones	Tom	1, 4
3	Smith	Jim	4, 5
4	Jones	Jim	2, 3, 5
5	Johnson	Susan	3, 4

- This is a fairly common beginner’s mistake.
- Putting multiple data items into a cell, as is done in the FriendsIDs column here, is very bad database design.
- It makes querying the table very difficult.

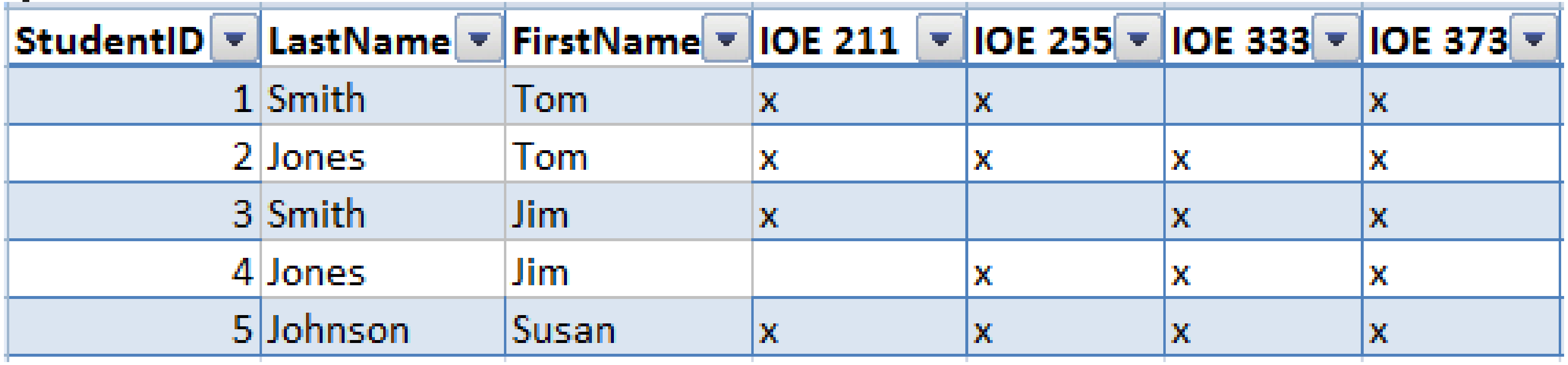
Putting Multiple Similar (Numbered) Fields in a Table Violates 1NF



StudentID	LastName	FirstName	Class1	Class2	Class3	Class4	Class5
1	Smith	Tom	IOE 211	IOE 333	IOE 373	ME 214	
2	Jones	Tom	IOE 333	IOE 373	AERO 101	PSYCH 235	
3	Smith	Jim	IOE 255	IOE 333	IOE 373		
4	Jones	Jim	EECS 211	ME 245	IOE 333	IOE 373	CHEM 341
5	Johnson	Susan	IOE 333	IOE 373			

- A very common mistake.
- Wastes space with all of the empty cells.
- And someone always comes along and signs up for six classes.
- And it’s hard to search! (Answering “Which students are taking IOE 373?” requires searching 5 columns.)

Having Multiple Fields That Are Similar Violates 1NF



StudentID	LastName	FirstName	IOE 211	IOE 255	IOE 333	IOE 373
1	Smith	Tom	x	x		x
2	Jones	Tom	x	x	x	x
3	Smith	Jim	x		x	x
4	Jones	Jim		x	x	x
5	Johnson	Susan	x	x	x	x

- This is a variation on the previous slide.
- Probably a little better, but still wrong.
- The course fields are all instances of the “course” entity. They don’t belong in a Students table.

So what’s wrong?

- The last three slides are examples of trying to represent one-to-many or many-to-many relationships in a single table.
- That violates 1NF, not to mention 3NF.
- BTW, to mention 3NF—a table must be in 1NF and 2NF before it has a possibility to be in 3NF.

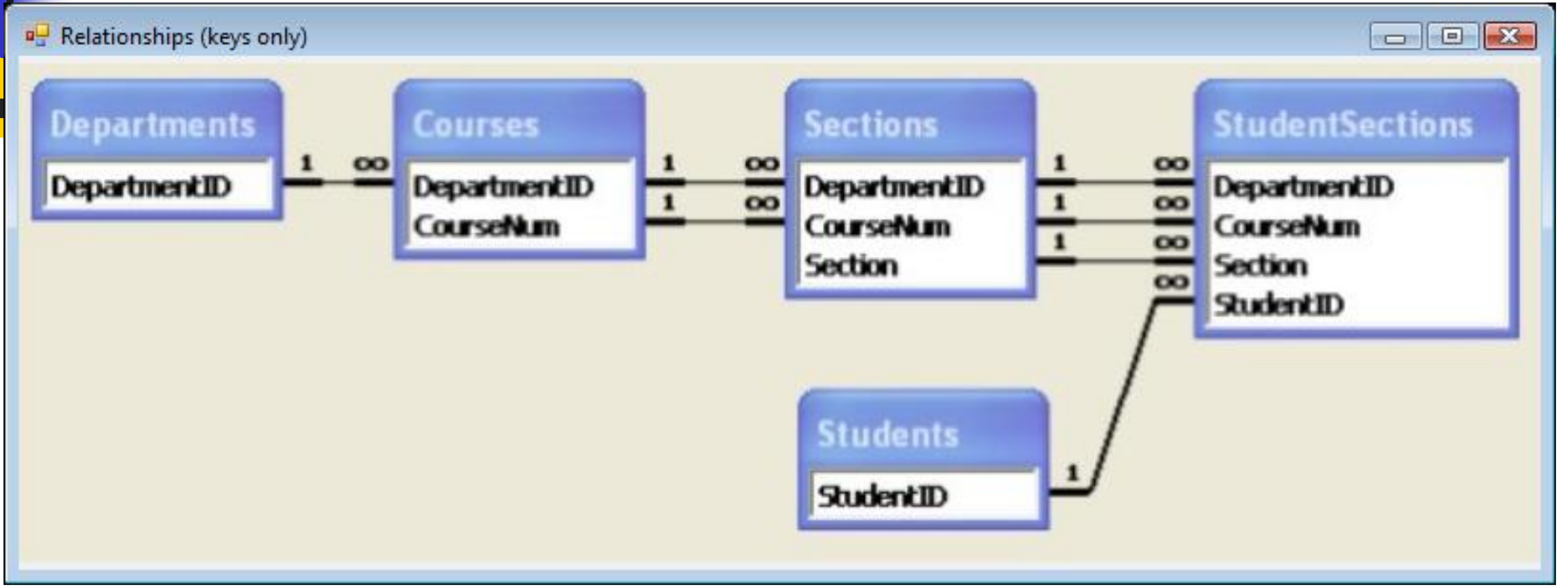
Second Normal Form (2NF)

- To be in 2NF, a table must:
  - Be in 1NF.
  - Have all non-key fields be attributes of the ENTIRE key.
  - 2NF applies only to compound keys; tables with simple keys that are in 1NF are usually in 2NF as well.

2NF

- 2NF is only an issue with tables having compound primary keys.
- 2NF is especially important when you have a chain of tables linked by relationships between their primary keys.
- The basic idea with 2NF is to put an attribute into the proper table—the table in which it is a property of the entire key, no more, no less.

Example: University



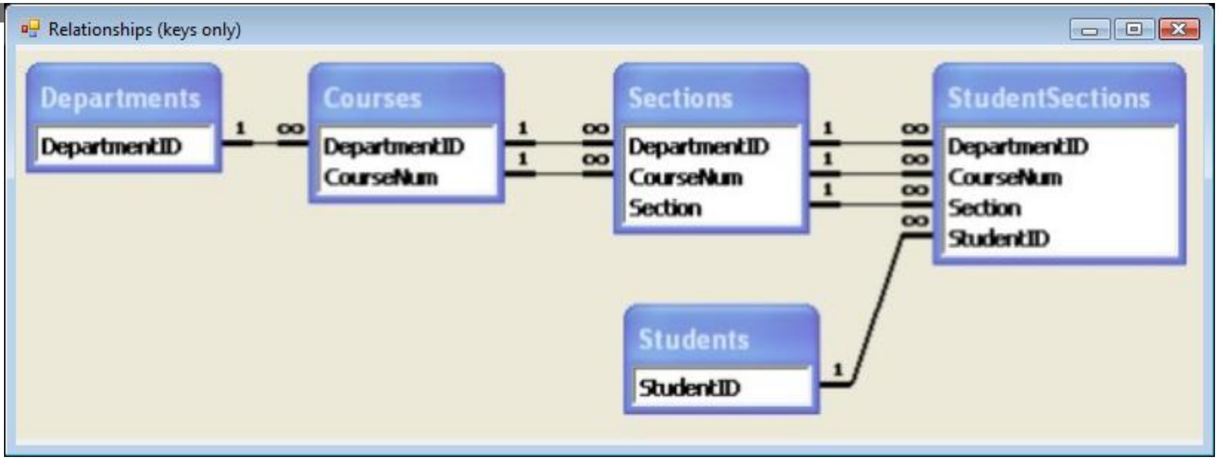
Non-key fields	Departments	Courses	Sections	StudentSections	Students
ClassTime	*DepartmentID	*DepartmentID	*DepartmentID	*DepartmentID	*StudentID
CourseDescription		*CourseNum	*CourseNum	*CourseNum	
DepartmentChair			*Section	*Section	
DepartmentName				*StudentID	
DepartmentOffice					
Grade					
InstructorID					
Major					
Prerequisite					
RoomNumber					
Uniqname					

- In which table should each non-key field go?

University: Solution

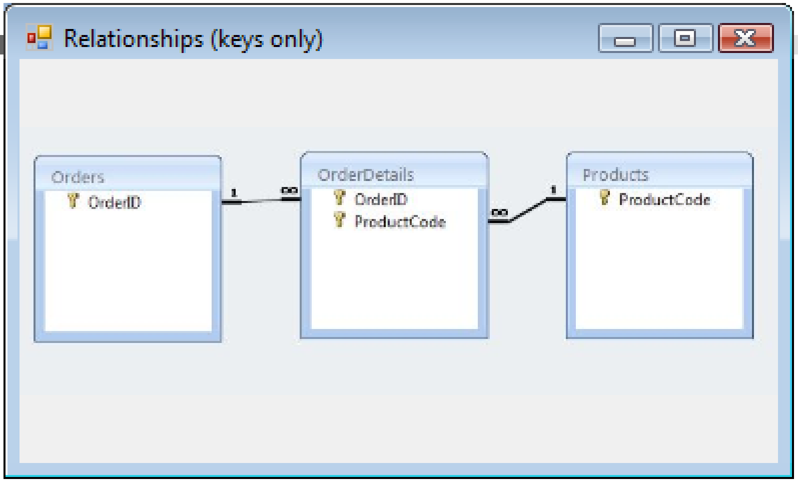
Non-key fields

- ClassTime
- CourseDescription
- DepartmentChair
- DepartmentName
- DepartmentOffice
- Grade
- InstructorID
- Major
- Prerequisite
- RoomNumber
- Uniqname



Departments	Courses	Sections	StudentSections	Students
*DepartmentID	*DepartmentID	*DepartmentID	*DepartmentID	*StudentID
DepartmentName	*CourseNum	*CourseNum	*CourseNum	Uniqname
DepartmentOffice	CourseDescription	*Section	*Section	Major
DepartmentChair	Prerequisite	InstructorID	*StudentID	
		RoomNumber	Grade	
		ClassTime		

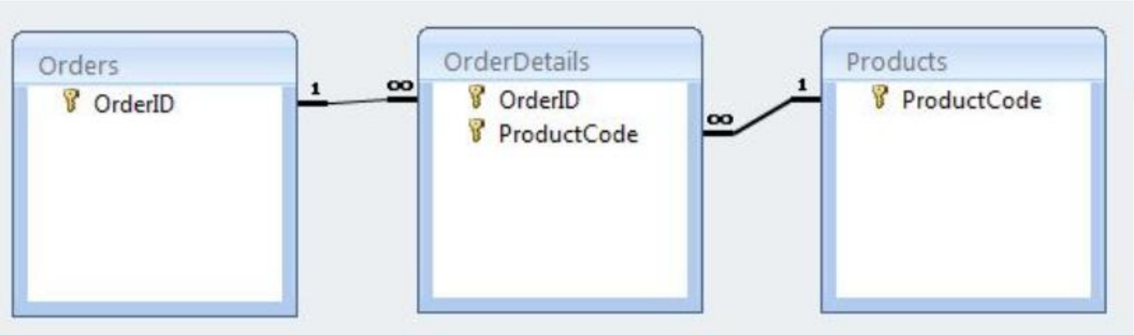
Example: Invoices



Non-key fields	Orders	OrderDetails	Products
Color	*OrderID	*OrderID	*ProductCode
Cost		*ProductCode	
CustomerID			
EmployeeID			
ProductName			
Quantity			
Size			

- Where should each non-key field go?

Invoices



Non-key fields
Color
Cost
CustomerID
EmployeeID
ProductName
Quantity
Size

Orders	OrderDetails	Products
*OrderID	*OrderID	*ProductCode
CustomerID	*ProductCode	ProductName
EmployeeID	Quantity	Color
	Cost	Size

- Issues:
  - If you don’t have different product codes for items that differ only in color or size, those two attributes could be included in OrderDetails instead.
  - Similarly, Cost depends on whether you have fixed prices (in which case cost belongs with Products), or if your prices change with time or if certain customers or orders get discounts.

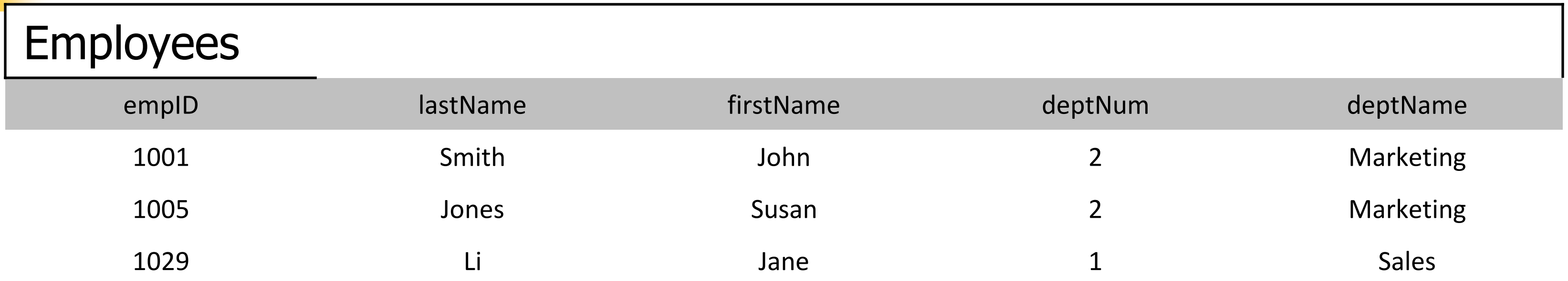
2NF Summary

- Second Normal Form is usually an issue that comes up mostly with compound keys (more than one field in the primary key).
- If a table is in 1NF and has an appropriate simple key it usually meets the requirements for 2NF.
- All non-key fields in a table with a compound key should be properties of the entire key (in combination)—not simply attributes of a part of the key.

Third Normal Form (3NF)

- 3NF is also fairly straightforward. A table in 3NF must be:
  - In 2NF; and
  - All attributes must be attributes of the key only; the table should not include attributes of attributes.
  - Attributes of attributes are technically referred to as “transitive dependencies”. Don’t put them in your tables!

Example



Employees

empID	lastName	firstName	deptNum	deptName
1001	Smith	John	2	Marketing
1005	Jones	Susan	2	Marketing
1029	Li	Jane	1	Sales

- Primary Key -> empID (1NF),
- If I know empID I have unique values for all other fields (2NF)
- So what’s the problem? deptNum and deptName; deptName is an attribute of deptNum (an attribute of an attribute), so we have a transitive dependency between deptNum and deptName!

Solution

Employees

empID	lastName	firstName	deptNum
1001	Smith	John	2
1005	Jones	Susan	2
1029	Li	Jane	1

Departments

deptNum	deptName
1	Sales
2	Marketing
3	Research

Remove the non-key fields that are dependent on a field that is not the primary key

The Golden Rule for 3NF

Every non-key field in a table should depend on
the key,
the whole key,
and nothing but the key,
so help me Codd.

Summary - Normalization

- Every entity should have a primary key (1NF)
- Every attribute should depend on the entire primary key (2NF)
- Every attribute should ONLY depend on the primary key (no transitive dependencies) (3NF)