

# Databases, Part I: Storing Information

In this section you will learn about: how information is stored in databases, different database relations, ways of ensuring data validity

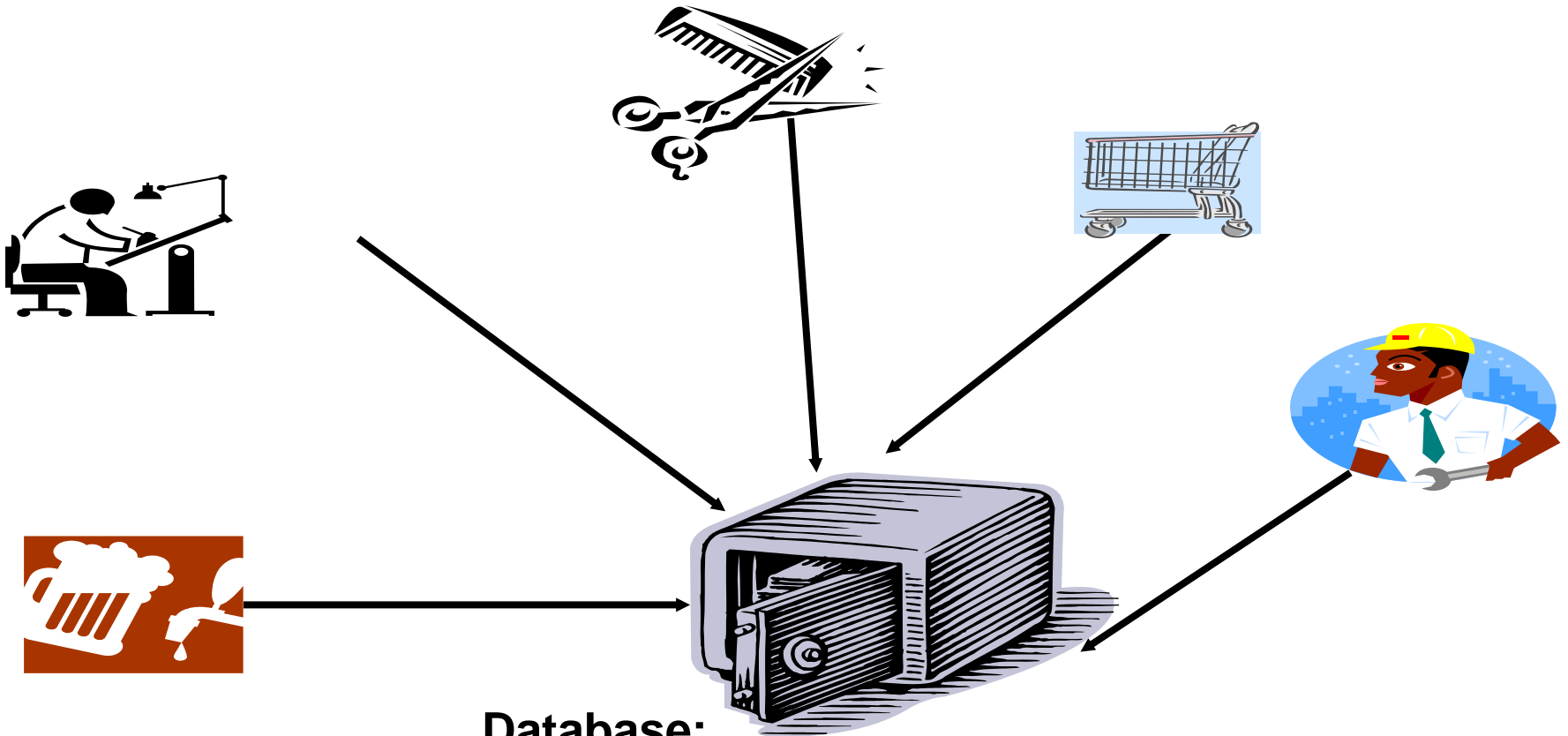
Online MS-Office information source:

<https://support.office.com/>

# Purpose Of A Database

- This section: To store information

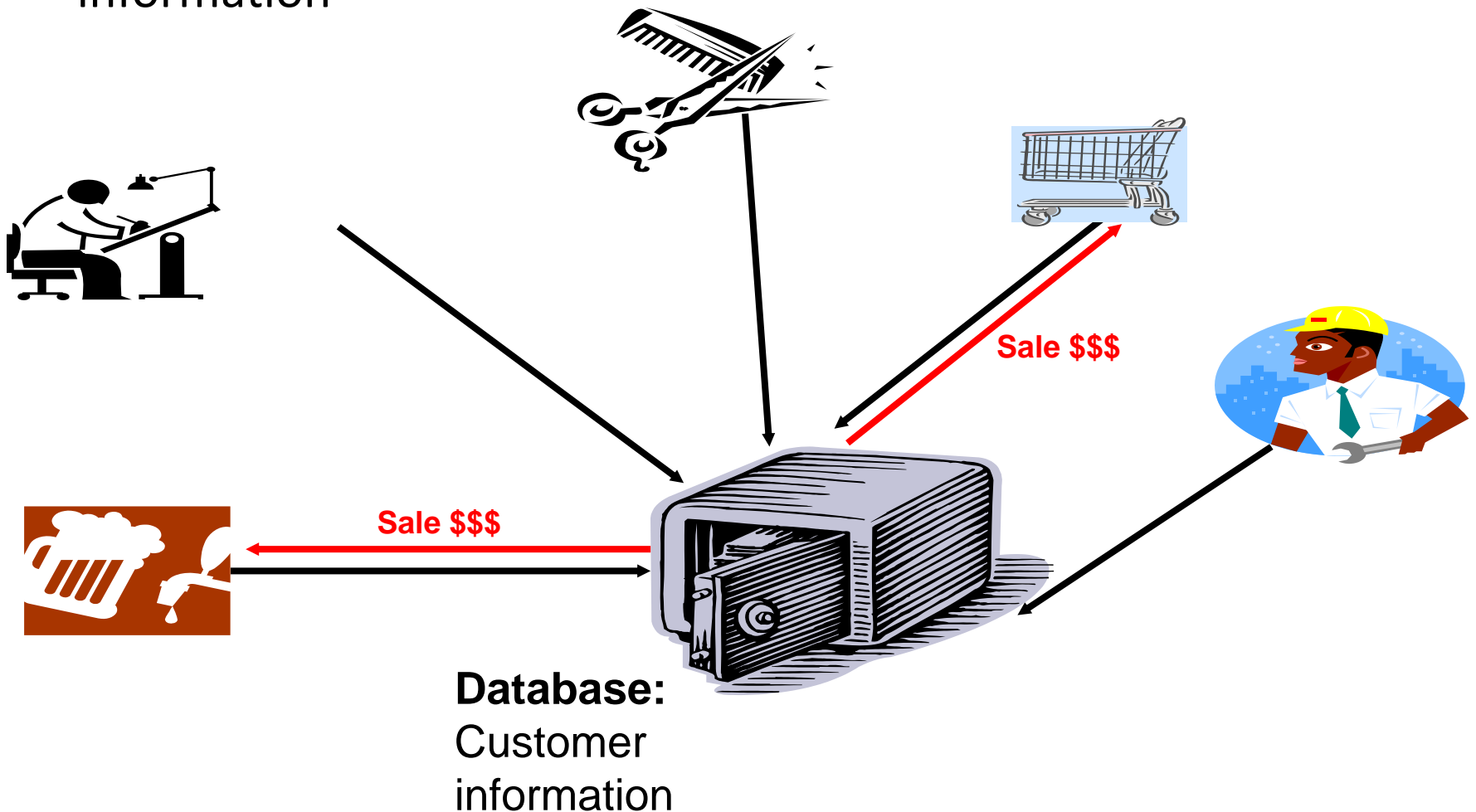
Francesco Rollandin/OpenClipart



**Database:**  
Customer  
information

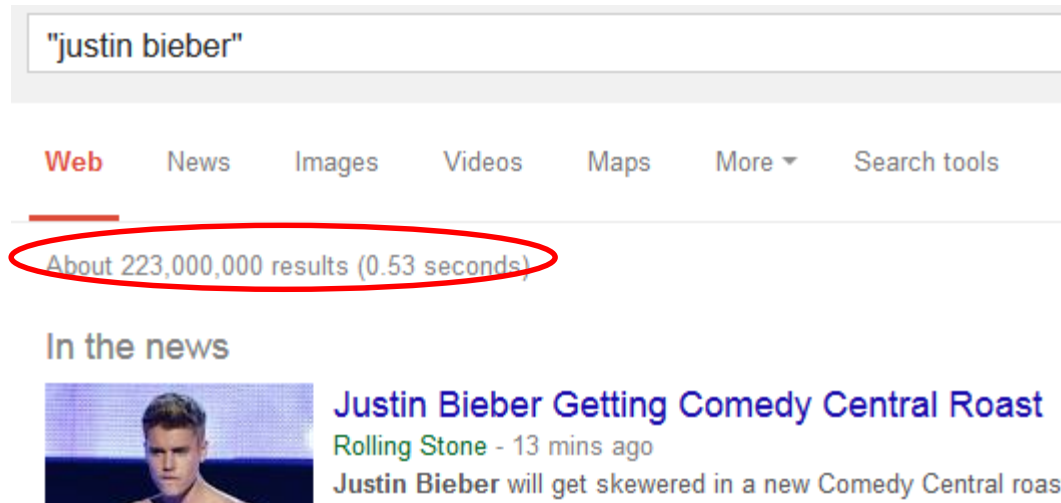
# Purpose Of A Database

- Next section (database queries): To retrieve information information



# Databases: Storing / Retrieving Information

- As you will see, implementing these two tasks aren't as easy as it seems.
- Information must be stored such that:
  - Information can be quickly retrieved




"justin bieber"

Web News Images Videos Maps More ▾ Search tools

About 223,000,000 results (0.53 seconds)

In the news

 **Justin Bieber Getting Comedy Central Roast**  
Rolling Stone - 13 mins ago  
Justin Bieber will get skewered in a new Comedy Central roast

# Databases: Storing / Retrieving Information (2)

- The database is designed to reduce problems during maintenance (additions, modifications, deletions)
  - Example: This comes up during database normalization (“if there is time”)



## Marketing Dept.

- Loren Coleman
- William McCloud



## Finance & Accounting

- Victor Davion
- Omiko Kurita

One employee  
has left and the  
whole  
department is  
gone?

# With Bother With Databases?

- Are used to store and retrieve information
- Why bother, why not use a simple file as an alternative?
  - E.g., tracking client information

## **MILES EDWARD O'BRIAN**

DS9 Corp

Electrical engineering

2007 purchases: \$10,000,000

2006 purchases: \$1,750,000

## **JAMIE SMYTHE**

Cooperative services

Gasoline refining

2006 purchases: \$5,000,0000

2005 purchases: \$5,000,0000

2004 purchases: \$5,000,0000

2003 purchases: \$5,000,0000

2002 purchases: \$5,000,0000

## **SCOTT BRUCE**

Bryce Consulting

Investment analysis

2007 purchases: \$500,000

2006 purchases: \$1,500,000

2005 purchases: \$2,500,000

2004 purchases: \$500,000

Etc.

- If the list is short then a simple text file may suffice
- As the list grows organizing and updating the information becomes more challenging (duplicates or inaccuracies?)
- Also searching the list according to specific criteria may become difficult
  - e.g., Show all clients whose purchases in 2007 were between one and five million dollars
  - e.g., Show all clients that made a purchase exceeding 10 million dollars.

# Storing Information In A Database

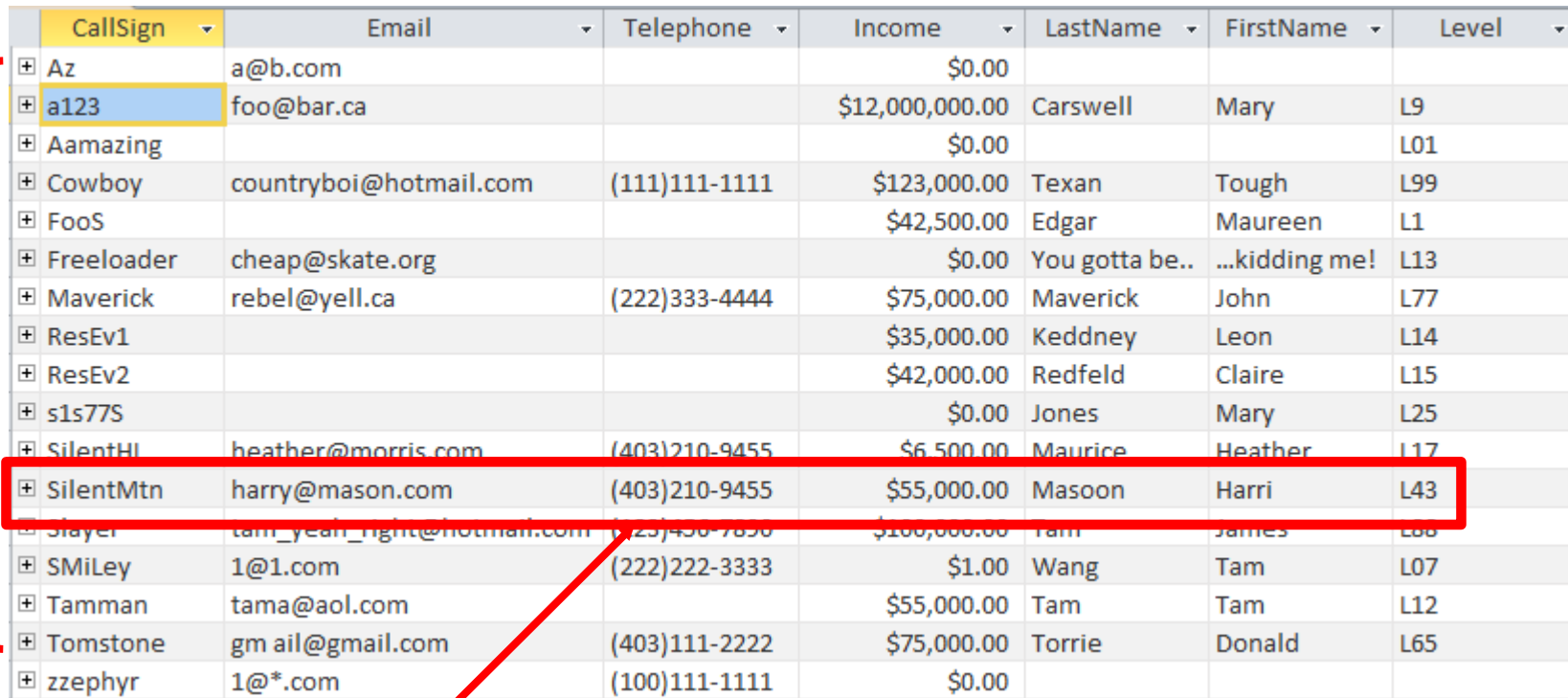
- Information is stored in tables:

## The 'Gamers' table

CallSign	Email	Telephone	Income	LastName	FirstName	Level
Az	a@b.com		\$0.00			
a123	foo@bar.ca		\$12,000,000.00	Carswell	Mary	L9
Aamazing			\$0.00			L01
Cowboy	countryboi@hotmail.com	(111)111-1111	\$123,000.00	Texan	Tough	L99
FooS			\$42,500.00	Edgar	Maureen	L1
Freeloader	cheap@skate.org		\$0.00	You gotta be..	...kidding me!	L13
Maverick	rebel@yell.ca	(222)333-4444	\$75,000.00	Maverick	John	L77
ResEv1			\$35,000.00	Keddney	Leon	L14
ResEv2			\$42,000.00	Redfeld	Claire	L15
s1s77S			\$0.00	Jones	Mary	L25
SilentHL	heather@morris.com	(403)210-9455	\$6,500.00	Maurice	Heather	L17
SilentMtn	harry@mason.com	(403)210-9455	\$55,000.00	Masoon	Harri	L43
Slayer	tam_yeah_right@hotmail.com	(123)456-7890	\$100,000.00	Tam	James	L88
SMiLey	1@1.com	(222)222-3333	\$1.00	Wang	Tam	L07
Tamman	tama@aol.com		\$55,000.00	Tam	Tam	L12
Tomstone	gm ail@gmail.com	(403)111-2222	\$75,000.00	Torrie	Donald	L65
zzephyr	1@*.com	(100)111-1111	\$0.00			

# Storing Information In A Database (2)

- Row = Record: An example instance of data within the table.
  - Gamers Table: one row is an example instance of a gamer



CallSign	Email	Telephone	Income	LastName	FirstName	Level
Az	a@b.com		\$0.00			
a123	foo@bar.ca		\$12,000,000.00	Carswell	Mary	L9
Aamazing			\$0.00			L01
Cowboy	countryboi@hotmail.com	(111)111-1111	\$123,000.00	Texan	Tough	L99
FooS			\$42,500.00	Edgar	Maureen	L1
Freeloader	cheap@skate.org		\$0.00	You gotta be..	...kidding me!	L13
Maverick	rebel@yell.ca	(222)333-4444	\$75,000.00	Maverick	John	L77
ResEv1			\$35,000.00	Keddney	Leon	L14
ResEv2			\$42,000.00	Redfeld	Claire	L15
s1s77S			\$0.00	Jones	Mary	L25
SilentHl	heather@morris.com	(403)210-9455	\$6,500.00	Maurice	Heather	L17
SilentMtn	harry@mason.com	(403)210-9455	\$55,000.00	Masoon	Harri	L43
Slayer	tam_yeah_tight@hotmail.com	(423)438-7838	\$100,000.00	Tam	James	L88
SMiLey	1@1.com	(222)222-3333	\$1.00	Wang	Tam	L07
Tamman	tama@aol.com		\$55,000.00	Tam	Tam	L12
Tomstone	gm ail@gmail.com	(403)111-2222	\$75,000.00	Torrie	Donald	L65
zzephyr	1@*.com	(100)111-1111	\$0.00			

One record, 'Harri Masoon'



# Storing Information In A Database (3)

- Column: are that attributes that we track for each record
  - Gamers Table: each column specifies the information we store about the gamers in this database.

## Attributes ('fields' in Access) of each record



CallSign	Email	Telephone	Income	LastName	FirstName	Level
Az	a@b.com		\$0.00			
a123	foo@bar.ca		\$12,000,000.00	Carswell	Mary	L9
Aamazing			\$0.00			L01
Cowboy	countryboi@hotmail.com	(111)111-1111	\$123,000.00	Texan	Tough	L99
FooS			\$42,500.00	Edgar	Maureen	L1
Freeloader	cheap@skate.org		\$0.00	You gotta be..	...kidding me!	L13
Maverick	rebel@yell.ca	(222)333-4444	\$75,000.00	Maverick	John	L77
ResEv1			\$35,000.00	Keddney	Leon	L14
ResEv2			\$42,000.00	Redfeld	Claire	L15
s1s77S			\$0.00	Jones	Mary	L25
SilentHL	heather@morris.com	(403)210-9455	\$6,500.00	Maurice	Heather	L17
SilentMtn	harry@mason.com	(403)210-9455	\$55,000.00	Masoon	Harri	L43
Slayer	tam_yeah_right@hotmail.com	(123)456-7890	\$100,000.00	Tam	James	L88
SMiLey	1@1.com	(222)222-3333	\$1.00	Wang	Tam	L07
Tamman	tama@aol.com		\$55,000.00	Tam	Tam	L12
Tomstone	gm ail@gmail.com	(403)111-2222	\$75,000.00	Torrie	Donald	L65
zzephyr	1@*.com	(100)111-1111	\$0.00			

# Primary Key

- Each table should typically have one attribute designated as the primary key:
  - The primary key must be guaranteed to be unique
  - It must uniquely identify one record from another

SIN	LastName	FirstName	Address	City	Province
638666670	Cartland	Douglas	1109, 4944 Dalworth Dr	Silent Hill	Alberta
456789123	Cartman	Eric	456 Lynchview Road	Southpark	Alberta
670380456	Edgar	Maureen	300, Lockinvar Road	Calgary	Alberta
456889123	Flanders	Ned	60 Evergreen Terrace	Springfield	Alberta
413754621	Kennedy	Leon	808, 4900 Wildman Ave	Racoon City	Alberta
456438624	Lemoy	Leonard	55 Logic Way	Vulcan	Alberta
666666667	Mason	Harry	7 Luckstone Dr	Silent Hill	Alberta
666666666	Morris	Heather	7 Luckstone Dr	Silent Hill	Alberta
444638047	Redfield	Claire	653 Wildpark Place	Racoon City	Alberta
123115323	Simcox	Cole	311 Ocean View Drive	Vancouver	British C
456789124	Simpson	Homer	59 Evergreen Terrace	Springfield	Alberta
123456789	Smith	John	123 Peanut Lane	Calgary	Alberta
666666668	Sunderland	James	7 Heartbroken Ave	Silent Hill	Alberta
620451097	Williams	Amanda	25 Rodeo Drive	Edmonton	Alberta
666666669	Wolf	Claudia	66 Twisted View	Silent Hill	Alberta
371988812	Carswell	Mary	425 Remington Ave	Calgary	Alberta

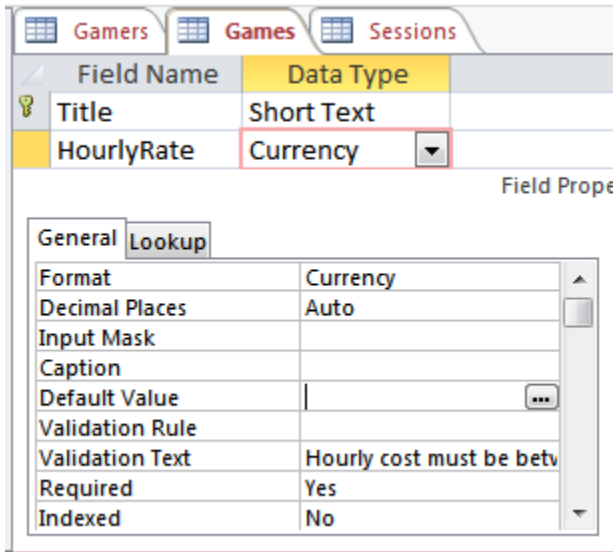
Primary Key  
for table  
'Employees'  
is the 'SIN'  
attribute

# Choosing A Primary Key

- A primary key must be unique to each record because it is the one thing that distinguishes them.
- If there's at least one instance where the attributes of two records can take on the same value that attribute cannot be a primary key. (When in doubt verify with your users).
- If a primary key cannot be formed from a single attribute then several attributes can be combined into a composite key. (Each attribute is still a column but together they form a unique primary key for each record).
  - E.g., CourseRegistrations table: Course name, course number, lecture section (CPSC 203 L01)
- If a unique primary key still cannot be found then 'invent' one.
  - E.g., StudentID#, SocialInsuranceNumber

# MS-Access: Views Of Your Database

- Design view



- Used to specify what attributes that a table will consist of:
  - e.g., GAMES: Title, HourlyRate
- Used to specify the type, format and valid range of values
  - e.g., SIN is attribute with 9 characters that must be in the format 000 000 000
  - e.g., HourlyRate must be between \$1 - \$100

- Datasheet view

Title	HourlyRate
DOOMED	\$7.00
EpicLegends	\$10.00
FarmerTam	\$6.00
FrankEsteinsHorror	\$15.00
GrecoAncients	\$20.00
LegendsOfLegend	\$5.00
MindBlowingLegends	\$20.00
Pirates	\$13.00
TheTams	\$20.00
WOWEE	\$10.00

- Once the attributes have been specified in the Design view using the Datasheet view allows data entry for each record.

# Example Problem: Online Games

- **This example can be found online:**
  - <http://pages.cpsc.ucalgary.ca/~tamj/2018/203W/database/LectureExample.accdb>
- An online gaming server will allow several online different games to be played
- Gamers can logon to play a particular game
- A gamer playing a game will create a 'session' that tracks (among other things) the cost of the gaming session
- (Even though some of the data stores information about games, it's really just a database example for a business where a product is utilized 'rental' for a finite period of time).

# Online Gamers: Information To Be Tracked

- Online identifier: *“Call sign”*
- Contact information: *Email*
- Contact information: *Telephone number*
- Income: *A (yearly) numeric figure*
- Real life identifier: *First and last name*
- Overall ‘score’ (sum of the player’s accomplishments among multiple games): *Level*

# Online Games: Information To Be Tracked

- Name of the game: *Title*
- The cost of playing a game: *Hourly rate*

# Gaming Sessions: Information To Be Tracked

- Each time a player starts playing a game, billing information must be generated (attaches the bill to the correct player)
  - Who played the game (who gets the bill)
  - Which game was played (how much is the cost per time unit)
  - How long was the game played (in conjunction with the cost per time unit it determines the amount for the bill)



# Picking Tables

- A table stores related information about an entity
  - E.g.,
  - Book: Title, author/authors, publisher, edition
  - Product: Product name, price, description
- The three groups of information (entities) in this problem appear to map to three database tables
  - Gamers
  - Games
  - Sessions

# Guidelines For Naming Tables<sup>1</sup>

1. Create a unique and descriptive name.
  - “CaloriesBurnedExercising” vs. “Workout”
2. Consider using the plural form of a name.
  - “Games table” vs. “Game table”
3. Avoid the use of spaces in names.
  - ~~“Undergraduate students”~~ vs. “Undergraduate\_Students” vs. “UndergraduateStudents”

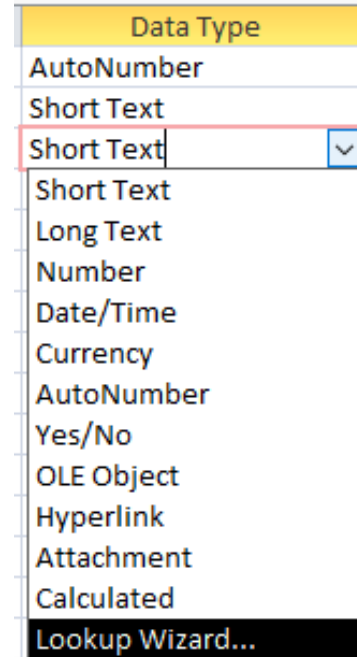
# Guidelines For Naming Attributes<sup>2</sup>

1. Create a name that accurately, clearly and unambiguously identifies the characteristic that the attribute represents.
  - “Name” vs. “FirstName”
2. Use the singular form of a name
  - Tables store multiple records (e.g., GAMES table), attributes store a single piece of information (e.g., Title for a particular game)
  - Do not fall into the pitfall of creating composite attributes (**phone numbers** - NO) vs. (home phone, cell phone etc. – YES)
3. Avoid the use of spaces in names (similar to tables).

# Type Of Data For An Attribute<sup>1</sup>

<sup>1</sup> Source (last accessed 2017): <https://support.office.com/> and the built in Office 2016 help system

- Most of the time you will select text
  - Short text: max of 255 characters
  - Long text: Up to 1 GB of characters (only the first 64,000 displayed)
  - Text allows data to be entered in specific format (e.g. for formatted postal codes, phone numbers).
    - Not to be used to constrain numeric ranges (e.g. age must be greater than 0).
- AutoNumber
  - Automatically generates a sequence of numbers 1,2,3..
  - Useful for generating unique primary keys if you cannot come up with one

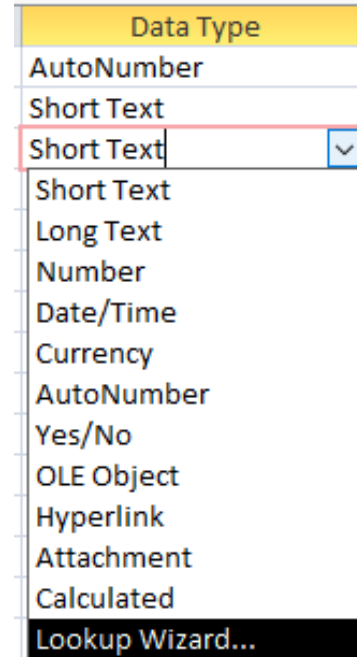


# Type Of Data For An Attribute(2)

- Obvious types: number, date/time, currency
  - The attribute must be set to date if you want set up error checking rules (“validation rules” – covered later in this section) by date e.g. When entering year of birth for “Generation Z” employees they must have a birth year of 1996 or later.
- Lookup wizard: when one table’s attribute refers to an attribute of another table (more on this later)

# More Advanced Types (If There Is Time)

- OLE (Object linking and embedding):
  - Allows 'objects' to be inserted e.g. MS-Office documents, images (similar to an email attachment)
- Hyperlink
  - Enter the web address, clicking on the attribute will automatically pull up the webpage in the default browser.
- Calculated
  - Similar to how one cell in Excel can be derived from the values in other cells
  - In Access 'calculated' allows for the attributes of a table to be mathematically calculated from the attributes of the same table (unless otherwise told explicitly don't use this for the assignment).



# Null Values

- Refers to the attributes of a record that are empty
- Primary keys cannot be null but other attributes may be null
- Entry of any attribute can be made mandatory (if data must be entered then it cannot be null)

The image shows a screenshot of Microsoft Access. At the top, there is a tab labeled 'Gamers'. Below it is a table structure view with two columns: 'Field Name' and 'Data Type'. The table has four rows: 'CallSign' (Text), 'Email' (Text), 'Telephone' (Text), and 'Income' (Currency). The 'Telephone' row is highlighted in yellow. Below the table structure is the 'Field Properties' window, which is currently showing the 'Lookup' tab. The 'Required' property is set to 'No', which is circled in red. Other properties shown include 'Field Size' (255), 'Format', 'Input Mask' (\(999\)999\.-0000), 'Caption', 'Default Value', 'Validation Rule', 'Validation Text', and 'Allow Zero Length' (Yes).

Field Name	Data Type
CallSign	Text
Email	Text
Telephone	Text
Income	Currency

Field Properties	
Lookup	
Field Size	255
Format	
Input Mask	\(999\)999\.-0000
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	Yes

# Gamers Table: Attributes

- Gamer information to track:
  - Online identifier: *“Call sign”*
  - Contact information: *Email*
  - Contact information: *Telephone number*
  - Income: *A (yearly) numeric figure*
  - Real life identifier: *First and last name*
  - Overall score: *Level*

## GAMERS

CallSign	Email	Telephone	Income	LastName	FirstName	Level



# Games Table: Attributes

- Game information to track:
  - Name of the game: *Title*
  - The cost of playing a game: *Hourly rate*

## GAMES

Title	HourlyRate

# Sessions Table: Attributes

- Each time a player plays a game billing information must be generated.
  - Who played the game
  - Which game was played
  - How long was the game played
- This one is trickier!
  - Identifying ‘who’: need to be 100% certain that the correct gamer has been identified (don’t bill the wrong person)
  - Identifying ‘which’: again certainty is required because different games have different hourly rates (don’t bill for the wrong game and/or generate a bill for an incorrect amount)
  - We need to “hold off” on creating a table until the above two requirements can be met

# Refinements Needed: Gamers

- Primary key?

## GAMERS

CallSign	Email	Telephone	Income	LastName	FirstName	Level

# Modified Table: Gamers

- Primary key: *CallSign*

## GAMERS

<u>CallSign</u>	Email	Telephone	Income	LastName	FirstName	Level

# Refinements Needed: Games

- Primary key?

## GAMES

Title	HourlyRate

# Modified Table: Games

- Primary key: **Title**

## GAMES

<u>Title</u>	HourlyRate

# The Games Table Again

- What if the game title was not guaranteed to be unique?
- Primary key?

# The Sessions Table Revisited

- Recall: Each time a player logs in to play a game, billing information must be generated.
- Some information need to generate a bill
  - Who played the game
  - Which game was played
- The ‘who’ needed to identify the gamer and the ‘which’ needed to specify the game
- Now that primary keys have been chosen for those two tables we can specify those two attributes (the primary keys unambiguously identify records from each table ‘who’, ‘which’)

## SESSIONS

CallSign	Title



# Foreign Key

- An attribute in one table that refers to an attribute in another table:
  - E.g. CallSign in the Sessions table actually refers to an actual players 'call sign' in the Gamers table
  - This is important because the CallSign is entered into the Gamers table and not into the Sessions table.

**GAMERS**

<u>CallSign</u>	Email	Telephone	Income	LastName	FirstName
Cowboy					

Retrieve from



**SESSIONS**

<u>CallSign</u>	Title
Cowboy	TheTams

Retrieve from

**GAMES**

<u>Title</u>	HourlyRate
TheTams	\$20

# Purpose Of Foreign Keys

- Using foreign keys can prevent errors
- Example: when we create a login playing session, we can ensure that we only bill a player that already exists in the Gamers table.

## Creating a new session

	TheTams	Cowboy	9/14/2015	1
	TheTams	Tamman	9/13/2015	120
	FarmerTam	<input type="text" value=""/>		0
*		a123		0
		Cowboy		
		FooS		
		Freeloader		
		Maverick		
		ResEv1		
		ResEv2		
		s1s77S		
		SilentHL		
		SilentMtn		
		Slayer		
		SMiLey		
		Tamman		
		Tomstone		

## Gamers Table

CallSign	
a123	f
Cowboy	c
FooS	
Freeloader	c
Maverick	r
ResEv1	
ResEv2	
s1s77S	
SilentHL	h
SilentMtn	h
Slayer	t
SMiLey	l
Tamman	t
Tomstone	e

- (The same principle applies to the 'Title' foreign key)

# Refinements Needed: Sessions

- It's determined that each player can only login once per day (our client informs us about this limitation on usage)
- Players can login and play over multiple dates
- For each session we could store the login date and the duration (minutes):

## SESSIONS

CallSign	Title	SessionDate	SessionDuration
Cowboy	TheTams	9/13/2015	120

# Refinements Needed: Sessions

- Each row in the table is created when a gamer logs in on a particular date
- Primary key?

## SESSIONS

CallSign	Title	SessionDate	SessionDuration
Cowboy	TheTams	9/13/2015	120

# Composite Key

- Reminder: It's a primary key that consists of multiple attributes (multiple columns in a database table)

<u>Attribute1</u>	<u>Attribute2</u>	Attribute3	Attribute4

# Modified Table: Sessions

- Primary key (composite): CallSign, Title, Date
- The creation of the primary key 'makes sense' intuitively for this example based on the previous restrictions.

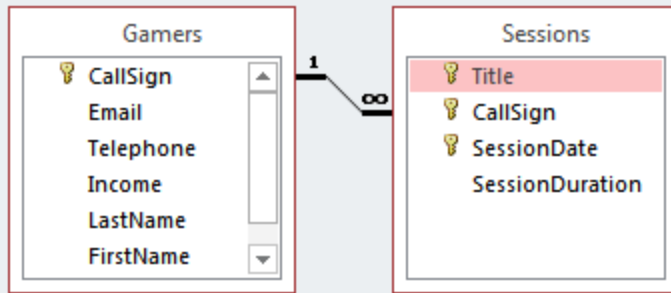
## SESSIONS

<u>CallSign</u>	<u>Title</u>	<u>SessionDate</u>	SessionDuration
Cowboy	TheTams	9/13/2015	120

- What would be the primary key if a player could login multiple times in a day?

# Relationships Between Tables

- Relationships occur when an attribute of one table is a foreign key in another table.



- Multiplicity: indicates how many instances of a particular item participates in the relationship:
  1. One to one
  2. One to many
  3. Many to many

# Multiplicity

## 1. One to one relationships

- One entity participates in the relationship from the ‘left’ and one entity participates in the relationship from the ‘right’.
- Person : Head
- Gamers : CallSign
- This type of relationship is *rare* in databases e.g. “DepartmentHead” and “Department” vs. “SIN” and “Employee”

## 2. One to many relationships

- On one side of the relationship one entity participates in the relationship while on the other side: zero or more entities may participate in the relationship.
- This is the *typical* type of database relation
- Person : Hair
- Gamers : Sessions : Games



# Multiplicity (2)

## 3. Many to many relationships

- On each side of the relationship zero or more entities may participate in the relationship.
- E.g., Travelers : Destinations
- A theoretical database relationship, *not directly implemented*

### Travelers table

TravelerID	LastName	FirstName
1	Tam	James
2	Jones	Mary
3	Smith	Jon

### Destinations table

DestinationID	DestinationName
1	Dubai
2	Paris
3	Cairo
4	Vulcan

# Multiplicity (3)

- Many to many relationships
  - Typically implemented as two one to many relationships in databases:

## Travelers table

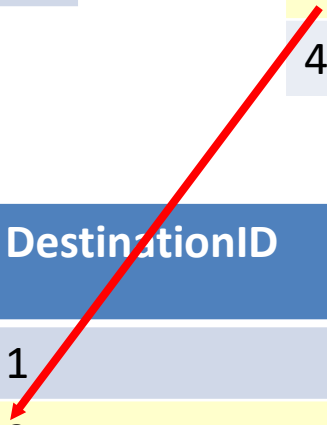
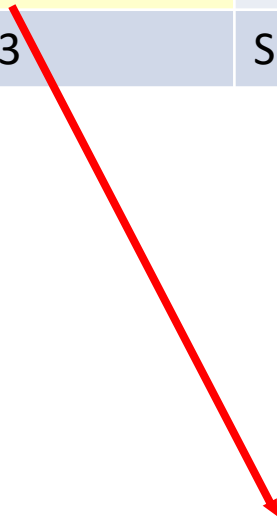
TravelerID	LastName	FirstName
1	Tam	James
2	Jones	Mary
3	Smith	Jon

## Destinations table

DestinationID	DestinationName
1	Dubai
2	Paris
3	Cairo
4	Vulcan

## Trips table

TravelerID	DestinationID	Date
1	1	Sept 1 2015
2	3	Sept 1 2015
2	4	Sept 8 2015



# Many To Many: Ignoring The Rule

## Travelers table

TravelerID	LastName	FirstName
1	Tam	James
2	Jones	Mary
3	Smith	Jon

Dest <sup>1</sup>	Dest <sup>2</sup>	Dest <sup>3</sup>	...	Dest <sup>n</sup>
Dubai				
Dubai	Cairo	Vulcan		Zimbobway
NY	Vulcan			

# Many To Many: Ignoring The Rule (2)

## Destinations table

DestinationID	DestinationName
1	Dubai
2	Paris
3	Cairo
4	Vulcan

Trav <sup>1</sup>	Trav <sup>2</sup>	Trav <sup>3</sup>	...	Trav <sup>n</sup>
Alice	Bob	Bill		Zeek
Alice	Bill	Charlie		
Alice	Bill			
Jim	Karen			

(Gamers : Games) could be implemented as a many to many relationship (by-passing the Sessions table) but problems similar to the previous example would be encountered.

# Primary-Foreign Keys Again

- When there is a one to many relationship the primary key of the 'one' side becomes a foreign key on the 'many' side.
- Examples:

<b>1</b>		<b>Many</b>
– Gamers	:	Sessions
<b>CallSign:</b>		<b>CallSign:</b>
<b>Primary key</b>		<b>Foreign key</b>

<b>1</b>		<b>Many</b>
– Games	:	Sessions
<b>Title:</b>		<b>Title:</b>
<b>Primary key</b>		<b>Foreign key</b>

This should make intuitive sense: the primary key uniquely identifies a record so it 'should' be on the 'one' rather than the many side

# Diagrammatically Representing Database Tables

- Entity-Relation diagrams (E-R Diagrams or E.R.D.s): show the attributes of a table

## Format

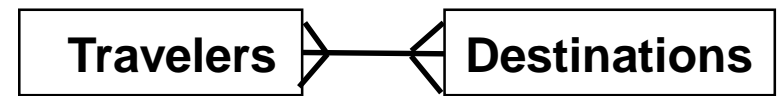
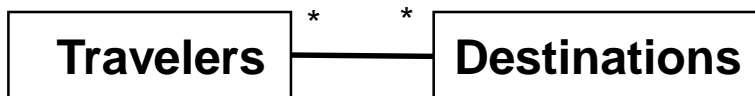
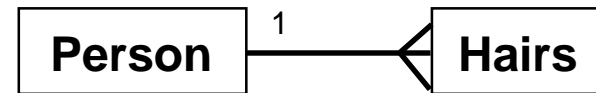
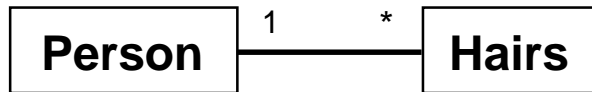
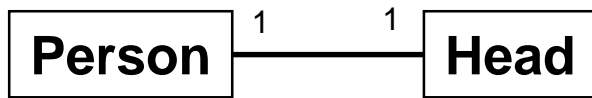
<b>TABLE NAME</b>
<u>Primary key</u>
Attribute
Attribute

## Example

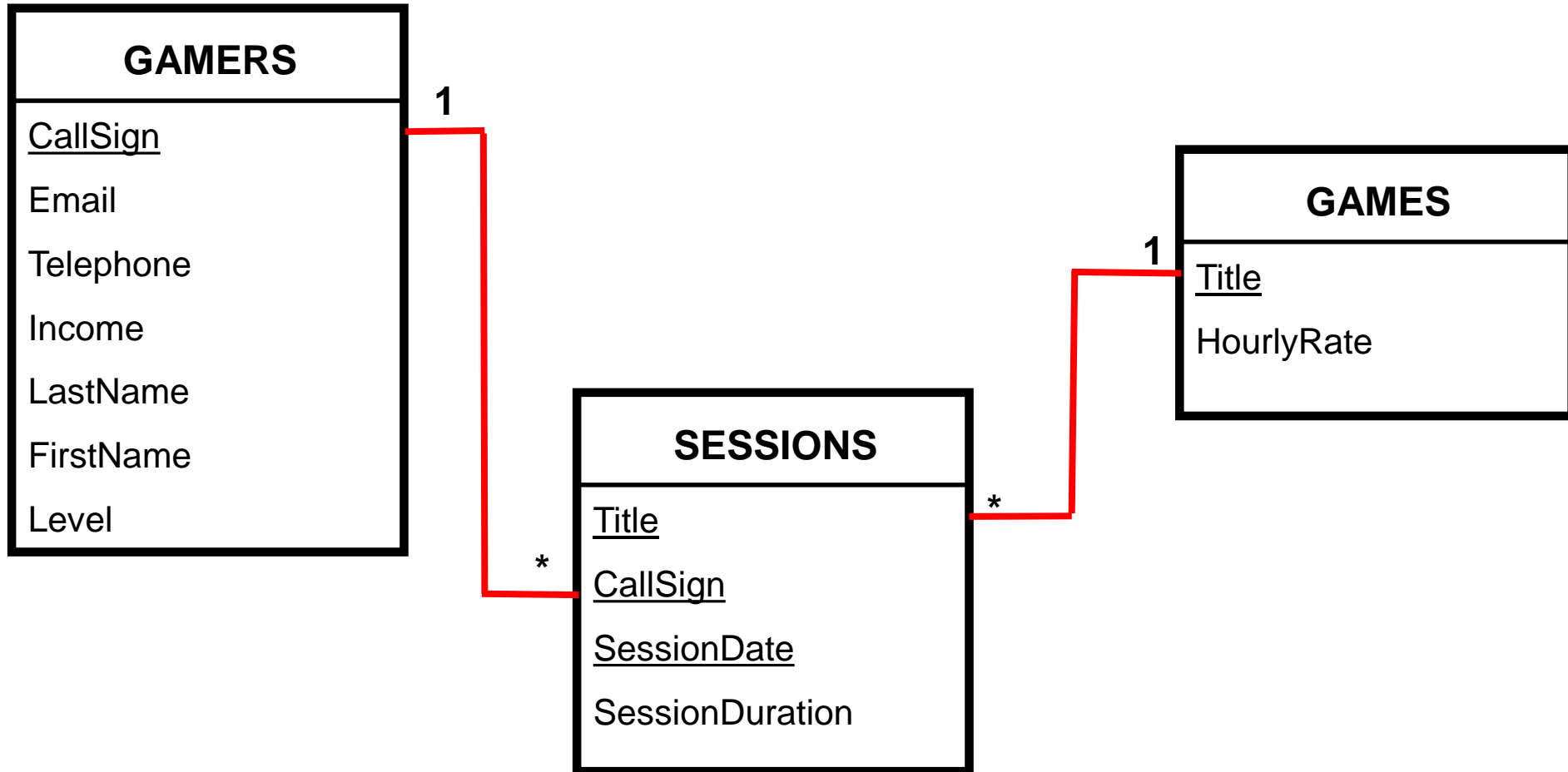
<b>GAMES</b>
<u>Title</u>
HourlyRate

# Diagrammatically Representing Relationships

- ERDs Graphically represent relationships between tables as well as any enforced rules on multiplicity:



# The ERD For The Example Database



- Note: the line specifying relationships between tables goes from the primary key to the foreign key (e.g. 'CallSign' in 'Gamers' to 'CallSign' in 'Sessions')



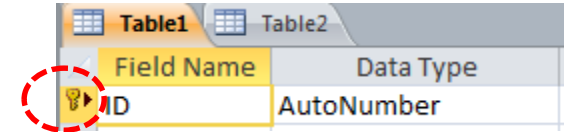
# Data Integrity

- High level description: prevent errors
- Some uses (there are others)
  - **Type checking:** prevent the wrong type of information from being entered e.g. alpha instead of numeric
  - **Range checking:** prevent information outside the acceptable range from being entered (e.g. negative age)
  - **Format checking:** prevent information in the wrong form from being entered (e.g. postal code N0N0N0 vs. N0N-0N0 vs. N0N 0N0)

# Types Of Data Integrity In Databases

## 1. Table-level integrity (entity integrity):

- Ensuring that no duplicate records exist.
- Implementation: no primary keys are null: MS-Access (automatic indexed – no duplicates).

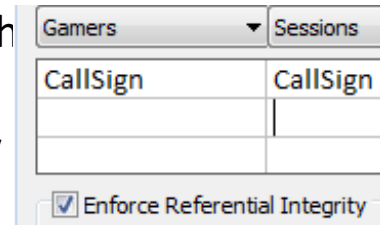


The screenshot shows a table structure with two columns: 'Field Name' and 'Data Type'. The first row contains 'ID' and 'AutoNumber'. A red dashed circle highlights the 'ID' field, and a yellow arrow points to it, indicating it is the primary key.

Field Name	Data Type
ID	AutoNumber

## 2. Relationship-level integrity (referential integrity):

- Ensuring that relationship between a pair of tables is sound and the records in the tables are synchronized when data is entered into, updated in or deleted from either table (MS-Access: only partially implemented).
- Partial implementation in Access: use 'lookup' for the 'data type' of an attribute & enforcing referential integrity.



The screenshot shows a relationship diagram between two tables: 'Gamers' and 'Sessions'. The 'CallSign' field in the 'Gamers' table is linked to the 'CallSign' field in the 'Sessions' table. The 'Enforce Referential Integrity' checkbox is checked.

Gamers	Sessions
CallSign	CallSign

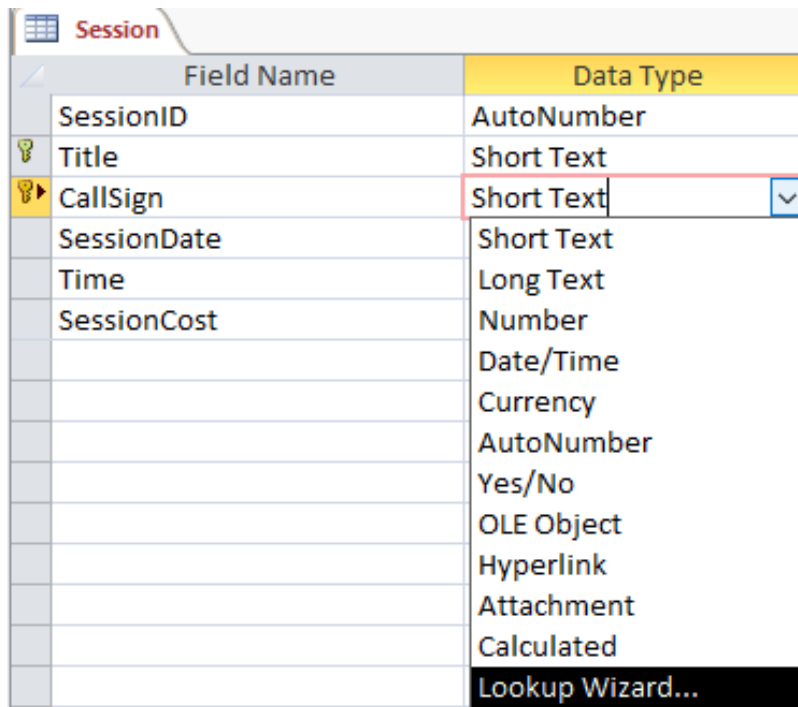
Enforce Referential Integrity

## 3. Field/attribute -level integrity (domain integrity):

- Ensuring that the attributes are valid and accurate (the previous slides 3 examples)
- MS-Access implementation: input masks and validation rules.

## 2. Relationship Level Integrity

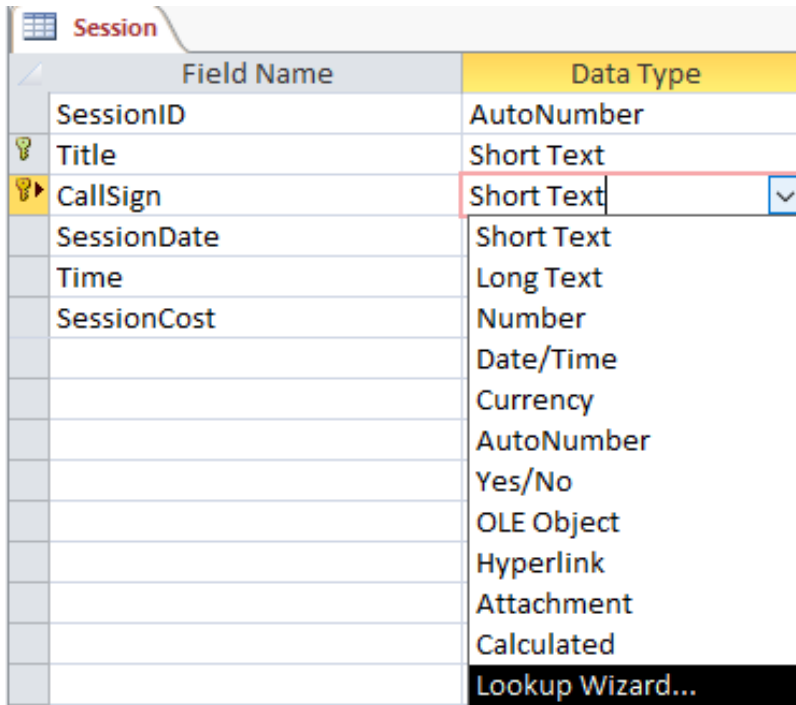
- Create the foreign-primary key relationship (Design view)



Field Name	Data Type
SessionID	AutoNumber
Title	Short Text
CallSign	Short Text
SessionDate	Short Text
Time	Long Text
SessionCost	Number
	Date/Time
	Currency
	AutoNumber
	Yes/No
	OLE Object
	Hyperlink
	Attachment
	Calculated
	Lookup Wizard...

# Relationship Level Integrity: Creating The Relationship

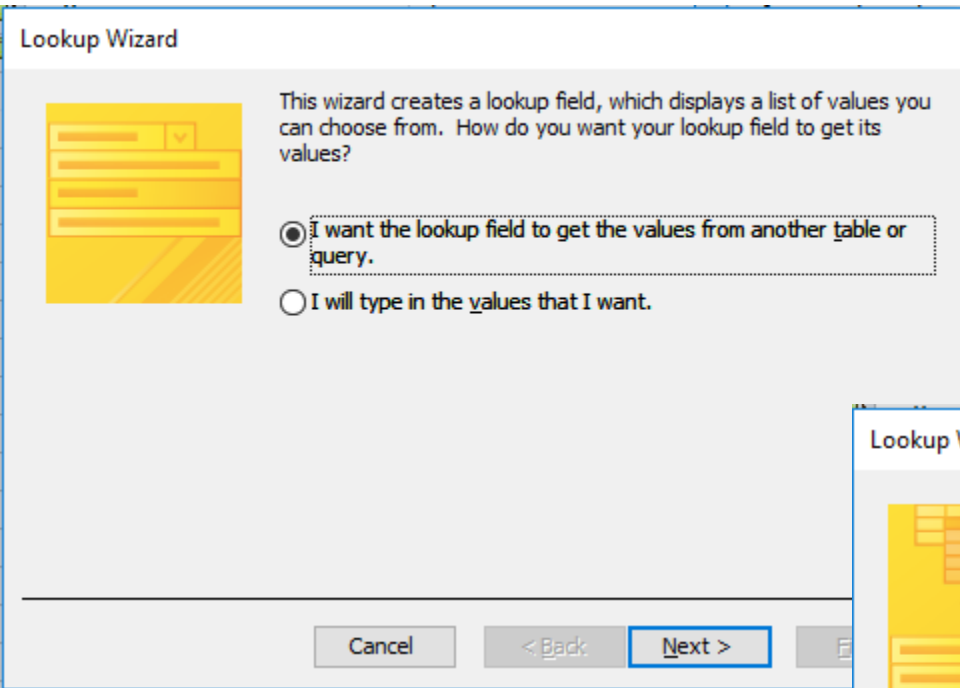
- Create the foreign-primary key relationship (Design view)



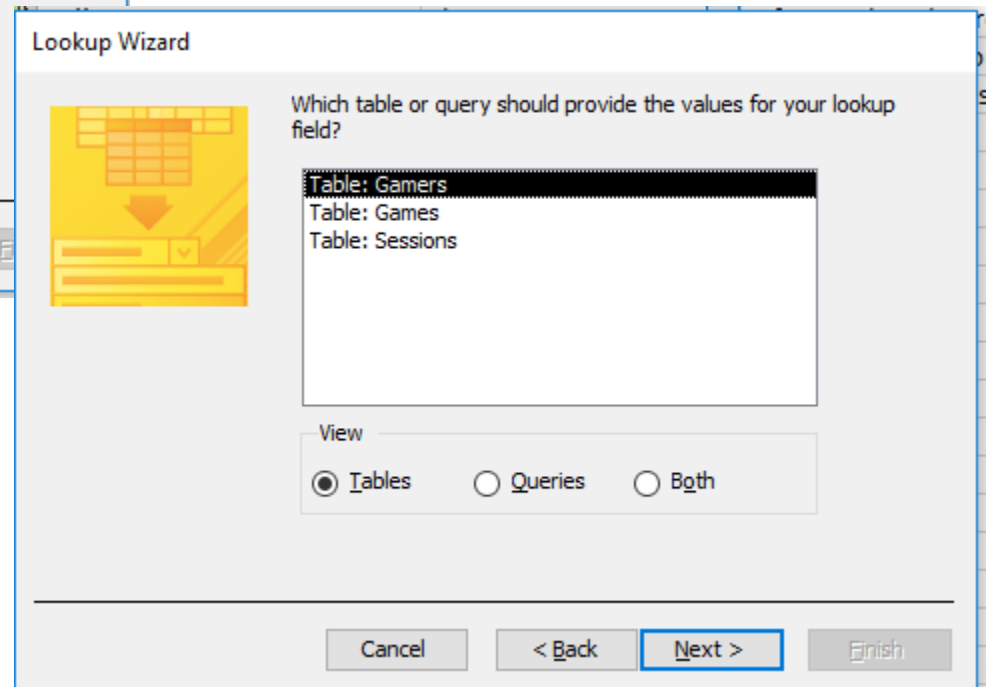
Field Name	Data Type
SessionID	AutoNumber
Title	Short Text
CallSign	Short Text
SessionDate	Short Text
Time	Long Text
SessionCost	Number
	Date/Time
	Currency
	AutoNumber
	Yes/No
	OLE Object
	Hyperlink
	Attachment
	Calculated
	Lookup Wizard...

# Relationship Level Integrity: Creating The Relationship (2)

Specify that the lookup value will come from another table



Specify the table (for CPSC 203)




# Relationship Level Integrity: Creating The Relationship

## (3)

Specify the foreign-primary key  
(attribute being looked up).

Lookup Wizard

Which fields of Gamers contain the values you want included in your lookup field? The fields you select become columns in your lookup field.



Available Fields:

- Email
- Telephone
- Income
- LastName
- FirstName
- Level

Selected Fields:

- CallSign

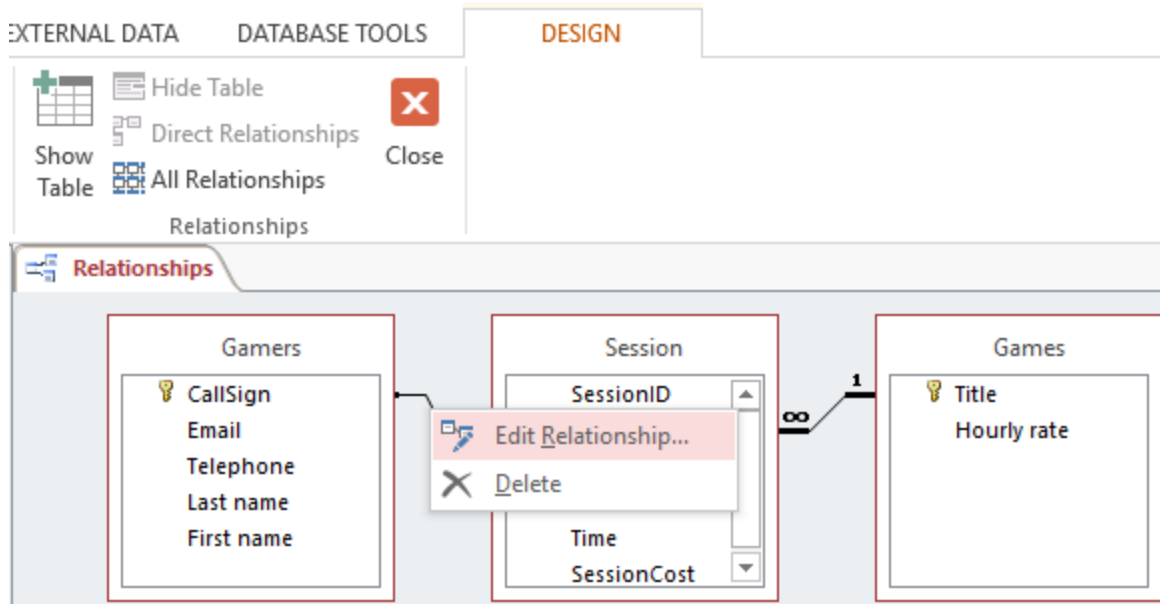
Buttons: Cancel, < Back, Next >, Finish

### Later sub-steps

1. Click next.
2. No need to sort the results
3. Adjust column width if you wish

# After Creating The Relationship

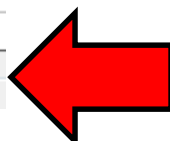
- You may edit the relationship



- Strengthen the relationship: **“Enforce referential integrity”**

Gamers	Sessions
CallSign	CallSign

Enforce Referential Integrity



## JT's Note

- Set up the relationship between tables as soon as possible.
- That's because other parts of the database may not work properly if set up the relationship afterwards.
- For this class make sure that you 'Enforce referential integrity'

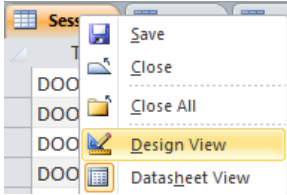


# Input Masks

- Ensures the proper format for the data entered into the database
- Example: SIN number must be entered as:
  - *<three digits> <space> <three digits> <space> <three digits>*
  - Invalid inputs:
    - *Abc def ghi*
    - *321 22 4234*
- Online example: Telephone number format
  - *(<area code>)<3 digits>-<4 digits>*
  - Example:
    - *(403)210-9455*

# Defining Input Masks

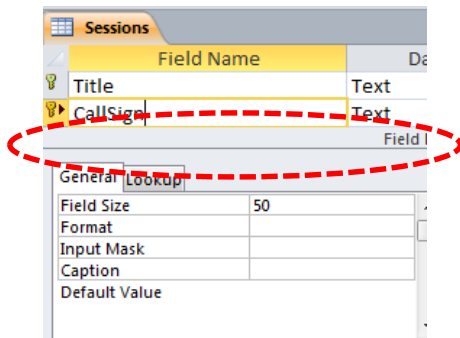
- Switch to 'design view'



- (The data type needs to be 'short text' which is the default)

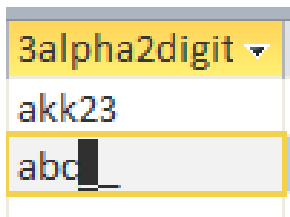
Field Name	Data Type
ID	AutoNumber
Mobile	Short Text

- Specify the required format under the 'Input mask' property of the appropriate table attribute

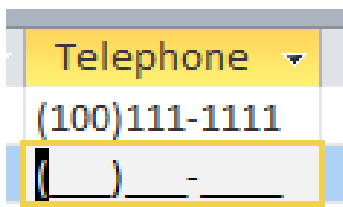


# Use Of Input Masks

- How it works: Constrains input allowed
  - Can only enter a single digit
  - Can only enter a single character
  - Can only enter 5 digits (zip code)
  - Etc.
- Benefits:
  - “Ignores” invalid inputs in real-time



- Specifies the format of data to be entered (data entry cues)



# Input Mask Codes

- Source (last accessed Sept 2015):
  - <https://support.office.com/>

Desired input	Character to enter as the input mask
A digit (0...9) <i>can</i> be entered	9
A digit (0...9) <i>must</i> be entered	0
Digits, space (default – data entry skipped), plus or minus sign	#
Alphabetic letter <i>must</i> be entered	L
Alphabetic letter <i>can</i> be entered	?
Alphabetic letter or digit <i>must</i> be entered	A
Alphabetic letter or digit <i>can</i> be entered	a
Converts characters that follow to <i>upper case</i>	>
Converts characters that follow to <i>lower case</i>	<

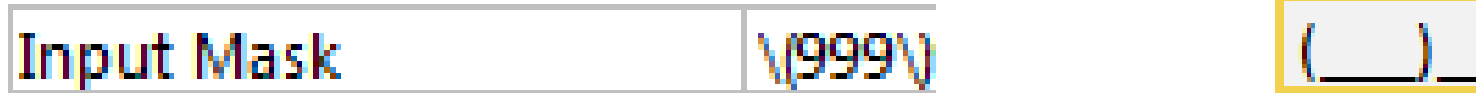
# Input Masks: Online Database Example

- Gamers table, level: always displays with 'L' at the beginning and then followed by one or two digits

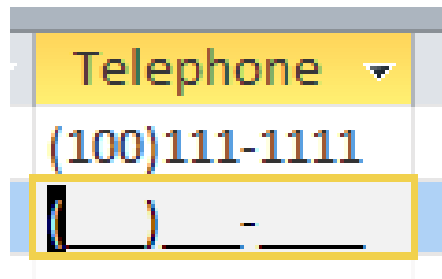
Level
L9
L01
L_

# Characters That Are Displayed But Not Part Of The Table Attributes

- Entering a slash '\ ' into the input mask (design view) will display a character in the datasheet (data entry) view

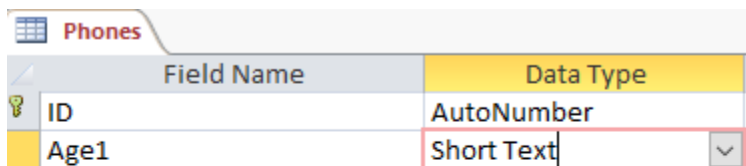


- This can be a helpful data entry/formatting cue
  - e.g. phone (*area code*)*digits-digits*

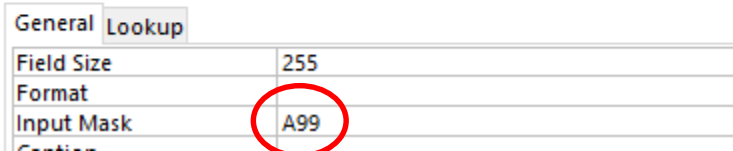


# Characters That Are Displayed But Not Part Of The Table Attributes (2)

- Note: the characters followed by a slash are NOT saved into the field of the database table
- Example

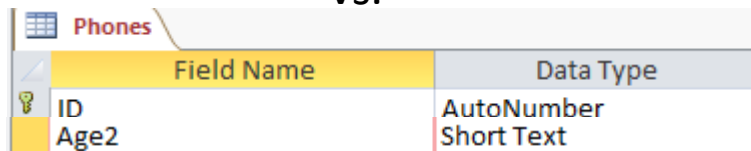


Field Name	Data Type
ID	AutoNumber
Age1	Short Text

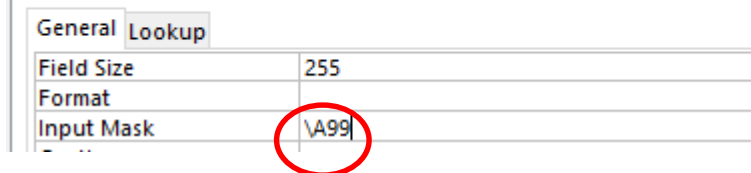


Property	Value
Field Size	255
Format	
Input Mask	A99

Vs.

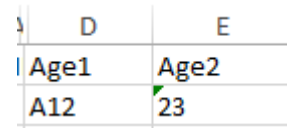


Field Name	Data Type
ID	AutoNumber
Age2	Short Text



Property	Value
Field Size	255
Format	
Input Mask	\A99

- Note: the data for the 'A' is saved for 'Age1' but not for 'Age2'



D	E
Age1	Age2
A12	23

- This can make a significant difference when later searching the database 'queries'
  - 'A12' can show up as a result for 'Age1'
  - 'A23' will not show up as a result for 'Age2'

# Multiple Slashes = Quotes

- If multiple “slash characters” (along with other characters) are used in immediate succession then Access will replace them with double quotes
  - This can be a handy shortcut

Format	
Input Mask	\(4\0\3\)

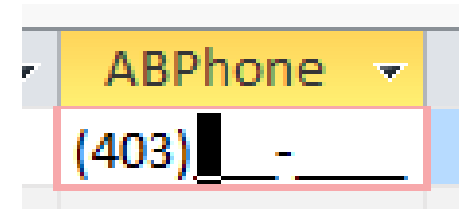
Format	
Input Mask	"(403)"



# Input Masks: Include The Slashes Or Not

- AGAIN: the character after the slash (or within the double quotes) will be displayed when the record is entered in the datasheet view.

Input Mask    "(403)"000-0000



The screenshot shows a data entry form with a dropdown menu set to "ABPhone". Below the dropdown is an input field containing the text "(403) -". The input field is highlighted with a red border, and a black cursor is visible at the end of the text.

- Benefits
  - A handy reminder of the format and type of data being entered
  - Reduces the need for repetitive data entry (i.e. if always the same for each record why require that it's entered each time) and reduces data entry errors (typos)
- Drawback:
  - AGAIN: the character after the slash (within the quotes) are not part of the attribute and cannot be entered or searched
  - E.g. all phone numbers in the above example must display with a 403 area code but you cannot search for 403 area codes.

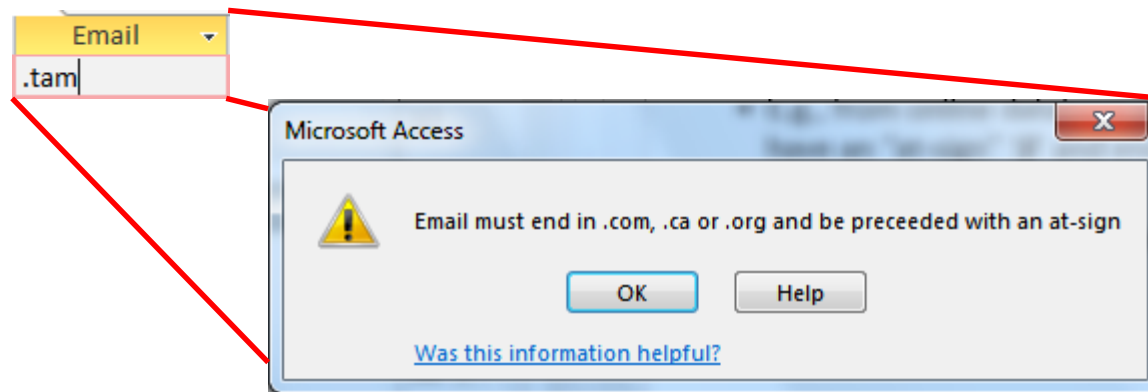
# Input Masks: Online Database Example

- Gamers table, telephone number: bracketed 3 digit area code, 3 digits, dash, 4 digits

Telephone ▼
(222)333-4444

# Validation Rules

- Validation rules check the data is in the valid range.
  - E.g., from online database example: Gamers table, income must be a non-negative value
- Can also be used to specify a data format (format of a “character string”)
  - E.g., from online database example: Gamers table, a valid email must have an “at-sign” ‘@’ and end in one of the following suffixes ‘.ca’, ‘.com’, “.org”
  - Unlike input masks validation rules allows useful error messages to be displayed



# Validation Rules: Specifying Error Messages

- “Validation text” & “default values”

The image shows a screenshot of the Microsoft Access interface. On the left, the 'Field Properties' window is open, with the 'Lookup' tab selected. The 'Validation Text' property is set to 'Balance cannot be zero or less'. A red arrow points from this text to a table below. The table has columns 'ID', 'Balance', and 'Click to Add'. The second row has 'ID' 2 and 'Balance' 0. A red arrow points from this row to a dialog box on the right. The dialog box is titled 'Microsoft Access' and contains a yellow warning icon, the text 'Balance cannot be zero or less', and 'OK' and 'Help' buttons. A blue link at the bottom says 'Was this information helpful?'.

ID	Balance	Click to Add
1	\$1.00	
2	0	
*	(New)	\$1.00

Microsoft Access

Balance cannot be zero or less

OK Help

[Was this information helpful?](#)

# Example Database: Application Of The Validation Rules

- Gamers table
  - CallSign: first character must be alphabetic
  - Email: must include an 'at-sign' = @ and then end in: '.ca', '.com' or '.org'
  - Income: no negative values
- Games table
  - HourlyRate: a dollar value from \$1 to \$100.
- Sessions table
  - SessionDate: date must be from Sept 12 2015 onwards
  - SessionDuration: specifies the number of seconds in the range of 0 – 86,400

# Validation Rules: Online Database Example (Single Range)

- Gamers table: Income (non-negative only)

Income	▼
\$12,000,000.00	
\$0.00	

# Validation Rules & Logic

- Logic can combine the conditions specified in validation rules

- AND (common)

- All conditions must be met before the data is deemed as valid

- Format:**

- (Condition1) And (Condition 2)

- Example:**

- $\geq 0$  And  $\leq 118$

- OR (rare for numeric ranges more common for character strings)

- At least one condition must be met before the data is deemed as valid

- Format:**

- (Condition1) Or (Condition 2)

- NOT (rare in databases)

- Format:**

- Not (Condition)

# Validation Rules: Online Database Example (Two Ranges)

- Games table: HourlyRate (a dollar value \$1 - \$100)

Title	HourlyRate
DOOMED	\$7.00
EpicLegends	\$10.00
FarmerTam	\$6.00
FrankEsteinsHorror	\$15.00
GrecoAncients	\$20.00
LegendsOfLegend	\$5.00
MindBlowingLegends	\$20.00

**Student exercise:** Sessions table, SessionDuration: Specifies the number of seconds from 0 – 86,400



# Validation Rules: Online Database Example (Date Ranges)

- Sessions: SessionDate: date must be from Sept 12 2015 onwards:
  - The date must be enclosed in a “number sign” pair #<date>#

Title	SessionDate
DOOMED	2015-09-16
DOOMED	2015-09-17
EpicLegends	2015-09-13
EpicLegends	2015-09-24
TheTams	2015-09-14
TheTams	2015-09-13

# Validation Rules: Specifying The Format Character Strings

- Character string: A sequence of characters (alpha, numeric and other characters) e.g. NX-01

Desired input	Value to enter into validation string	Example use
Alphabetic only (case insensitive)	[A-Z]	Like "[A-Z]" (single alpha) Like "[A-Z][A-Z]" (two alpha)
Digit only	[0-9] #	Like "[0-9]" (single digit) Like "[0-9][0-9]" (two digits) Like "###" (three digits)
Wildcards	* ?	Like "*" (anything) Like "?" (any <b>single</b> character)

# The **Wildcard**

- A value that can be used in place of other values.
- Example: “The joker is wild” option in card games
- Example: “\* **.docx**” only documents ending in the suffix “.docx” with any name will be considered.
- The start character ‘\*’ is a wildcard because it can be substituted by zero or more characters
  - Example documents that will be considered
    - resume.**.docx**
    - A.**.docx**
    - **.docx** fulfills the wildcard requirement but is not a valid filename.
  - Example documents that won’t be considered
    - resume.doc
    - Me.jpg
- The wildcard can be used in conjunction with validation rules

# Validation Rules: Online Database Example (Simple Character String)

- Gamers table: CallSign (first character must be alphabetic)

– O.K.

Tamman

Tomstone

zzephyr

– Not O.K.

1foo

# Validation Rules: Online Database Example (Complex Character String)

- Gamers table: Email (must contain an 'at-sign' in the string and the string ends with '.com', '.ca', '.org')

– O.K.

heather@morris.com

harry@mason.com

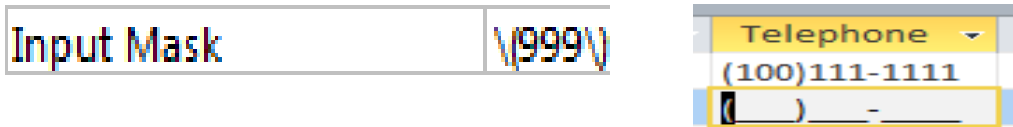
tam\_yeah\_right@hotmail.com

– Not O.K.

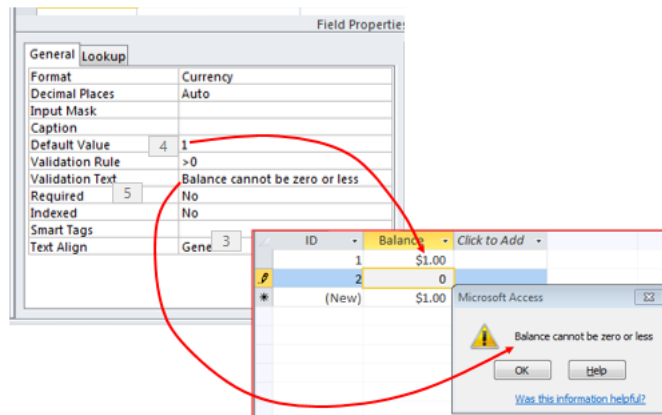
foo.com

# Input Masks Vs. Validation Rules: Error Handling

- Input masks
  - Can specify desired input beforehand, real-time error prevention



- Validation Rules
  - Default values can be specified
  - Customized and detailed error messages can be created
    - However, messages appear after erroneous data has been entered



# Input Masks Vs. Validation Rules: Error Handling

- Range checking e.g. age  $\geq 0$ 
  - Use a validation rule
- In general both can be used to check the format of the data
  - E.g. <digit><digit><alpha>
- Entering an arbitrary number of characters
  - Use a validation rule: Use of the multi-character wildcard (not possible using an input mask)

# Documenting A Database

- Documentation: Provides information about the database to the other people who will be working on database.
- In MS-Access documentation can be entered in the “Description” column (under the Design view)

Field Name	Data Type	Description
CallSign	Text	Must begin with an alphabetic character, the rest can be anything
Email	Text	<anything> @ <anything> end in .com, .ca or .org
Telephone	Text	Format: (ddd)ddd-dddd
Income	Currency	Must not be a negative value
LastName	Text	
FirstName	Text	
Level	Text	<L> <One or two digits> e.g., L1, L99

- It can provide information about the type and format of the information to be stored.
  - Can be used if errors are found. (Providing the original ‘intention’ if there is an error in the validation rules or the input mask can help others correct the error).



# Database Forms And Normalization (If There Is Time)

- A database form: design requirement of a database
- Forms discussed in this class:
  - First normal form (1NF), Second normal form (2NF), Third normal form (3NF)
  - Earlier forms (e.g. 1NF) are less strict than later forms (2NF)
  - Later forms (3NF) fulfill the requirements of earlier forms (2NF)
- Database normalization: redesigning a database in order to bring it from a less strict form to one that is more strict.

# Why Is Normalization Necessary?

- Normalization is regarded as good style
- My database 'works' that's "good enough" why bother?
- It also helps to prevent errors or problems which are caused by how the database is designed:
  - e.g., insertion anomalies: difficulties when adding new information
  - e.g., deletion anomalies: deleting information may result in the inadvertent loss of information

# Example Database Table: Projects<sup>1</sup>

<sup>1</sup> From “Database Development for Dummies” by Allen G. Taylor

- This table shows:
  - ResearcherID: each professor working on a research project is given a computer generated login name.
  - Research project: name of the projects worked on in a particular department.
    - Professors can work on multiple projects
    - Research projects can be initiated without a professor
  - Location: room number of the research lab.

ResearcherID (PK)	Research projects (PK)	Location
aturing	Graph Coloring	QC-103
	Traveling Salesman	QC-201
rdescartes	Knapsack	QC-121
cbabbage	Traveling Salesman	QC-201
	Knapsack	QC-121
bowen	Knapsack	QC-121

# Problem: Some Cells Can Contain Multiple Entries

- Queries can be awkward to form
  - E.g., Using the 'Like' operator is difficult because it must deal with special cases (or more entries in each cell).
  - Example:

Research projects
Graph Coloring
Traveling Salesman
Knapsack
Traveling Salesman
Knapsack
Knapsack

**With this format searching for projects under "Knapsack" won't work correctly (some labs show up with others will not).**

# Databases In First Normal Form

- **1NF.:** Each cell can contain *at most* one element (one value or a null value, the latter for non-primary key fields).
- The previous table in first normal form:

ResearcherID (PK)	Research project (PK)	Location
aturing	Graph Coloring	QC-103
aturing	Traveling Salesman	QC-201
rdescartes	Knapsack	QC-121
cbabbage	Traveling Salesman	QC-201
cbabbage	Knapsack	QC-121
bowen	Knapsack	QC-121

# First Normal Form: Critique

- **Improvements:**
  - Cells contain only one value which reduces some of the problems associated with forming queries.
- **Further improvements needed:**
  - There is redundancy in the table e.g., “aturing”

ResearcherID	ResearchProject	Location
aturing	Graph Coloring	QC-103
aturing	Traveling Salesman	QC-201

- It may be subject to modification (addition and deletion) anomalies.

# Deletion Anomaly

- Allan Turing (“aturing”) no longer works on the “Graph Coloring” project.

## Before

Researcher ID	Research Project	Location
aturing	Graph Coloring	QC-103
aturing	Traveling Salesman	QC-201
rdescartes	Knapsack	QC-121
cbabbage	Traveling Salesman	QC-201
cbabbage	Knapsack	QC-121
bowen	Knapsack	QC-121

## After

Researcher ID	Research Project	Location
aturing	Traveling Salesman	QC-103
rdescartes	Knapsack	QC-121
cbabbage	Traveling Salesman	QC-201
cbabbage	Knapsack	QC-121
bowen	Knapsack	QC-121

# Insertion Anomalies

- A new research project 'UFO' is added to the department and room 'Area-57' is to be used as the research lab but a researcher has not been hired.
- This is an incomplete record that cannot yet be properly added to the database (PK = researcher and project name)

ResearcherID	Research project	Location
aturing	Graph Coloring	QC-103
aturing	Traveling Salesman	QC-201
rdescartes	Knapsack	QC-121
cbabbage	Traveling Salesman	QC-201
cbabbage	Knapsack	QC-121
bowen	Knapsack	QC-121



# Problem With This Table

- The 'Projects' table combines two related but separate concepts:
  - Which research project a particular researcher working on
  - What is the location of a particular project

ResearcherID	Research project	Location
aturing	Graphic Coloring	QC-103
aturing	Traveling Salesman	QC-201

- It's a sign that a single unique key cannot be assigned
- By itself this isn't necessarily a problem (i.e., 'ResearcherID' and 'Research project' form a composite primary key).
- But the non-primary key element "Location" depends only on a part of the primary key ("Research project") which can lead to anomalies.

# Databases In Second Normal Form

- Every non-primary key element must be dependent on the primary key (and the entire primary key if the key is composite).
- The previous table split into two tables that are each in second normal form.

## ResearchProject

ResearcherID	Project
aturing	Graph coloring
rdescartes	Knapsack
cbabbage	Traveling Salesman
bowen	Knapsack

## ResearchLocation

Project	Location
Graph coloring	QC-103
Knapsack	QC-121
Traveling Salesman	QC-201

# Critique Of Second Normal Form

- Dependencies can still exist that affects the database but in a slightly more subtle fashion.
- All non-key fields are dependent upon the primary key but some may be dependent in an indirect fashion.


# Example<sup>1</sup>: “SalaryRange” Table

ResearcherID	AcademicRank	RangeCode
eschroedinger	Full professor	4
pdirac	Associate professor	3
wheisenberg	Full professor	4
hbethe	Assistant professor	2
jwheeler	Adjunct professor	1

**Primary key**



**Non-key fields  
whose values are  
dependent on the  
primary key  
(second normal  
form)**



# The Example In 2<sup>nd</sup> Normal Form Are Still Subject To Some Anomalies

- Example Professor Dirac leaves the university.

## Before

ResearcherID	AcademicRank	RangeCode
eschroedinger	Full professor	4
pdirac	Associate professor	3
wheisenberg	Full professor	4
hbethe	Assistant professor	2
jwheeler	Adjunct professor	1

## After

ResearcherID	AcademicRank	RangeCode
eschroedinger	Full professor	4
wheisenberg	Full professor	4
hbethe	Assistant professor	2
jwheeler	Adjunct professor	1

# Problem With The Database (2<sup>nd</sup> Normal Form)

- While both non-key elements are dependent upon the primary key, with “RangeCode” that dependency is indirect.

ResearcherID	AcademicRank	RangeCode
eschroedinger	Full professor	4
pdirac	Associate professor	3

- “RangeCode” is dependent upon “AcademicRank” which is in turn dependent upon “ResearcherID”.
- This is referred to as a transitive dependency:

**RangeCode** → **AcademicRank** → **ResearcherID**

# Third Normal Form

- A database in third normal form fulfills the requirements of second normal form and has no transitive dependencies.
- Previous example in third normal form:

## ResearcherRank

ResearcherID	AcademicRank
eschroedinger	Full professor
pdirac	Associate professor
wheisenberg	Full professor
hbethe	Assistant professor
jwheeler	Adjunct professor

## RankRange

AcademicRank	Range Code
Full professor	4
Associate professor	3
Assistant professor	2
Adjunct professor	1

# After This Section You Should Now Know

- How a database is broken down into tables and how tables are broken down into its component parts
- What are the type of tables and the purpose of each
- What is the purpose of a primary key
- What is a foreign key
- When table are related what is the rule for determining which table contains the primary vs. foreign key
- What is a null value
- What are forms of data integrity in databases
- Guidelines for naming tables and the attributes of the tables
- What are the three relationships that may exist between tables and how they differ



## After This Section You Should Now Know (2)

- How is a many-to-many relationship typically implemented in a database
- The ERD representation of databases
- (If there is time): What are the characteristics of a database in: first normal form, second normal form, third normal form (if there is time)