



# Data Query Using MySQL

John Kim

Field Station Programs

San Diego State University



# talk outline

- Queries using SQL
- SQL by example
- Hands-on queries through PhpMyAdmin



# SQL

- Structured Query Language
- ANSI standard computer language with many variants (I'll use MySQL)
- Read, write, calculate, modify data
- Easy to learn, hard to master



# Querying Tips

- Build queries incrementally.
- Make a back-up copy of tables.
- Manual: <http://dev.mysql.com/doc/>



# Executing SQL Commands through PHPMyAdmin

Enter any SQL command and see results



Server: marsh.tinternet.edu Database: rcn\_admin Table: mytbl

Structure Browse **SQL** Search Insert Export Operations Emp

InnoDB free: 10240 kB

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	sdf	int(10)		UNSIGNED	No		auto_increment	

Check All / Uncheck All With selected:

Print view Relation view Propose table structure

Add 1 field(s)  At End of Table  At Beginning of Table  After sdf Go



# 4 Basic Queries:

- **SELECT** - retrieves data from a database table
- **INSERT** - inserts new data into a table
- **UPDATE** - updates data in a database table
- **DELETE** - deletes data from a database table



# SELECT

Basic Syntax: **SELECT** *columns* **FROM** *table*

- 1) get specific columns from a table:

```
SELECT cover, height FROM observation
```

- 2) get all columns:

```
SELECT * FROM observation
```

- 3) get specific rows from a table:

```
SELECT cover, height FROM observation  
WHERE cover > 1
```



## SELECT (continued)

More complex conditions:

```
SELECT * FROM observation
```

```
WHERE ( height > 10 AND height < 20 ) OR cover > 10
```





## JOINS

```
SELECT * FROM observation
```

```
SELECT * FROM table1 LEFT JOIN table2  
ON match up two fields
```

```
SELECT * FROM observation LEFT JOIN species  
ON species_id = species_id
```

```
SELECT * FROM observation LEFT JOIN species  
ON observation.species_id = species.species_id
```



# LEFT OUTER JOIN aka LEFT JOIN

SPECIES_ID	COVER	HEIGHT
4	0.5	4
2	0.1	2
4	0.01	4
4	0.1	5
1	0.5	12
3	0.25	15

SPECIES_ID	NAME
0	LATR2
1	ERPU8
3	LEFE
4	GUSA2



SPECIES_ID	COVER	HEIGHT	SPECIES_ID	NAME
4	0.5	4	4	GUSA2
2	0.1	2		
4	0.01	4	4	GUSA2
4	0.1	5	4	GUSA2
1	0.5	12	1	ERPU8
3	0.25	15	3	LEFE



# INNER JOIN

SPECIES_ID	COVER	HEIGHT
4	0.5	4
2	0.1	2
4	0.01	4
4	0.1	5
1	0.5	12
3	0.25	15

SPECIES_ID	NAME
0	LATR2
1	ERPU8
3	LEFE
4	GUSA2



SPECIES_ID	COVER	HEIGHT	SPECIES_ID	NAME
4	0.5	4	4	GUSA2
2	0.1	2		
4	0.01	4	4	GUSA2
4	0.1	5	4	GUSA2
1	0.5	12	1	ERPU8
3	0.25	15	3	LEFE



# INSERT

Basic syntax:

**INSERT INTO** *table (list of fields)* **VALUES** (*list of values*)

**INSERT INTO** species (species) **VALUES** ('alin')

**INSERT INTO** location (site, web, plot, quad) **VALUES** ('P', 2, 'E', 1)



# UPDATE

Basic syntax: **UPDATE** *table* **SET** *field* = *value*

```
SELECT * FROM observation  
  WHERE comments = "NA"
```

```
UPDATE observation SET comments = "Not Applicable"  
  WHERE comments = "NA"
```

```
SELECT comments FROM observation  
  WHERE comments LIKE "never%"
```

```
UPDATE observation  
  SET comments = concat("Cannot identify as of ", curdate())  
  WHERE comments LIKE "never%"
```



# Update with multiple tables

Basic Syntax: **UPDATE** *tables* **SET** *assignments* **WHERE** *conditions*;

Useful for normalizing tables. After creating a separate Location table...

**UPDATE** observations, locations

**SET** observations.location\_id=location.location\_id

**WHERE** observation.site = location.site AND  
observation.web=location.web AND  
observation.plot=location.plot AND  
observation.quad=location.quad



# DELETE

Basic syntax:

**DELETE FROM** *table* **WHERE** *condition*

**DELETE FROM** species **WHERE** species = 'acer saccharum'

But good to try a select first, before deleting:

**SELECT FROM** species **WHERE** species = 'acer saccharum'



# Aggregate Functions

You can use function in place of plain column names.

```
SELECT AVG(height) FROM observation
```

Examples: AVG(), STDDEV(), VARIANCE(), MAX(), MIN()

Calculate the average height of sand muhly (MUAR2):

```
SELECT AVG(height) FROM observation  
LEFT JOIN species ON observation.species_id = species.species_id  
WHERE species.species = 'MUAR2'
```





# Normalizing An Existing Data Table

Overview:

NPP Table

Date	Site	Web	Plot	Quad	Species	Cover	Height	Comments
2/5/2005	G	1	N	1	BOER4	1	12	
2/5/2005	G	1	N	1	BOER4	0.5	12	
2/5/2005	G	1	N	1	BOER4	0.25	10	
2/6/2005	G	1	N	1	LATR2	9	24	grazed
2/6/2005	G	1	N	2	LATR2	5	18	
2/6/2005	G	1	N	2	SPCR	4	12	

- A. Make the location table
- B. Use it to assign foreign keys to observations table
- C. Delete location info in observations table

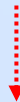


# Normalizing An Existing Data Table: Step 1

Make 2 copies of the original table

NPP Table

Date	Site	Web	Plot	Quad	Species	Cover	Height	Comments
2/5/2005	G	1	N	1	BOER4	1	12	
2/5/2005	G	1	N	1	BOER4	0.5	12	
2/5/2005	G	1	N	1	BOER4	0.25	10	
2/6/2005	G	1	N	1	LATR2	9	24	grazed
2/6/2005	G	1	N	2	LATR2	5	18	
2/6/2005	G	1	N	2	SPCR	4	12	



Observations

Date	Site	Web	Plot	Quad	Species	Cover	Height	Comments
2/5/2005	G	1	N	1	BOER4	1	12	
2/5/2005	G	1	N	1	BOER4	0.5	12	
2/5/2005	G	1	N	1	BOER4	0.25	10	
2/6/2005	G	1	N	1	LATR2	9	24	grazed
2/6/2005	G	1	N	2	LATR2	5	18	
2/6/2005	G	1	N	2	SPCR	4	12	

Locations

Date	Site	Web	Plot	Quad	Species	Cover	Height	Comments
2/5/2005	G	1	N	1	BOER4	1	12	
2/5/2005	G	1	N	1	BOER4	0.5	12	
2/5/2005	G	1	N	1	BOER4	0.25	10	
2/6/2005	G	1	N	1	LATR2	9	24	grazed
2/6/2005	G	1	N	2	LATR2	5	18	
2/6/2005	G	1	N	2	SPCR	4	12	

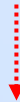


# Normalizing An Existing Data Table: Step 2

To **observations** table, add a primary key and a `location_id` field

Observations

Date	Site	Web	Plot	Quad	Species	Cover	Height	Comments
2/5/2005	G	1	N	1	BOER4	1	12	
2/5/2005	G	1	N	1	BOER4	0.5	12	
2/5/2005	G	1	N	1	BOER4	0.25	10	
2/6/2005	G	1	N	1	LATR2	9	24	grazed
2/6/2005	G	1	N	2	LATR2	5	18	
2/6/2005	G	1	N	2	SPCR	4	12	



Observations

ID	Location_id	Date	Site	Web	Plot	Quad	Species	Cover	Height	Comments
1		2/5/2005	G	1	N	1	BOER4	1	12	
2		2/5/2005	G	1	N	1	BOER4	0.5	12	
3		2/5/2005	G	1	N	1	BOER4	0.25	10	
4		2/6/2005	G	1	N	1	LATR2	9	24	grazed
5		2/6/2005	G	1	N	2	LATR2	5	18	
6		2/6/2005	G	1	N	2	SPCR	4	12	





# Normalizing An Existing Data Table: Step 4

Fill **locations** table by pulling distinct data from observations.

pulling distinct data:

```
SELECT distinct site, web, plot, quad FROM observations
```

pulling distinct data and putting it into locations table:

```
INSERT INTO locations SELECT distinct site, web, plot, quad  
FROM observations
```

Site	Web	Plot	Quad
FPC	1	E	1
FPC	1	E	2
FPC	1	E	3
FPC	1	E	4
FPC	1	N	1
FPC	1	N	2

Locations



# Normalizing An Existing Data Table: Step 5

To **locations** table, add a primary key. Set auto-increment.

ID	Site	Web	Plot	Quad
1	FPC	1	E	1
2	FPC	1	E	2
3	FPC	1	E	3
4	FPC	1	E	4
5	FPC	1	N	1
6	FPC	1	N	2

Locations



# Normalizing An Existing Data Table: Step 6

Update foreign key in **observations** table using **locations** table.

```
UPDATE observations, locations
SET observations.location_id=locations.id
WHERE observations.site=locations.site AND
observations.web=locations.web AND
observations.plot=location.plot AND
observations.quad=locations.quad
```

**Observations**

ID	Location_id	Date	Site	Web	Plot	Quad	Species	Cover	Height	Comments
1		2/5/2005	G	1	N	1	BOER4	1	12	
2		2/5/2005	G	1	N	1	BOER4	0.5	12	
3		2/5/2005	G	1	N	1	BOER4	0.25	10	
4		2/6/2005	G	1	N	1	LATR2	9	24	grazed
5		2/6/2005	G	1	N	2	LATR2	5	1	
6		2/6/2005	G	1	N	2	SPCR	4	1	

**Locations**

ID	Site	Web	Plot	Quad
1	FPC	1	E	1
2	FPC	1	E	2
3	FPC	1	E	3
4	FPC	1	E	4
5	FPC	1	N	1
6	FPC	1	N	2



# Normalizing An Existing Data Table: Step 7

In **observations** table, delete the location columns.

Observations

ID	Location_id	Date	Species	Cover	Height	Comments
1	24	2/5/2005	BOER4	1	12	
2	24	2/5/2005	BOER4	0.5	12	
3	24	2/5/2005	BOER4	0.25	10	
4	24	2/6/2005	LATR2	9	24	grazed
5	25	2/6/2005	LATR2	5	18	
6	25	2/6/2005	SPCR	4	12	

Locations

ID	Site	Web	Plot	Quad
1	FPC	1	E	1
2	FPC	1	E	2
3	FPC	1	E	3
4	FPC	1	E	4
5	FPC	1	N	1
6	FPC	1	N	2





# Normalizing Existing Data with MS Access

The problem: How to split up large data into smaller chunks with correct foreign keys?

Date	Site	Web	Plot	Quad	SPECIES	COVER	HEIGHT	COUNT	COMMENTS
2/2/2005	FPC	1	E	1	ERPU8	0.5	4	13	NA
2/3/2005	FPC	1	E	1	ERPU8	0.1	2	16	NA
2/4/2005	FPC	1	E	1	GUSA2	0.01	4	2	NA
2/5/2005	FPC	1	E	1	GUSA2	0.1	5	1	NA
2/6/2005	FPC	1	E	1	GUSA2	0.5	12	1	NA
2/7/2005	FPC	1	E	1	LEFE	0.25	5	1	NA

ID numbers need to match

Date	Site	Web	Plot	Quad	Species_id	COVER	HEIGHT	COUNT	COMMENT
2/2/2005	FPC	1	E	1	1	0.5	4	13	NA
2/3/2005	FPC	1	E	1	1	0.1	2	16	NA
2/4/2005	FPC	1	E	1	2	0.01	4	2	NA
2/5/2005	FPC	1	E	1	2	0.1	5	1	NA
2/6/2005	FPC	1	E	1	2	0.5	12	1	NA
2/7/2005	FPC	1	E	1	3	0.25	5	1	NA

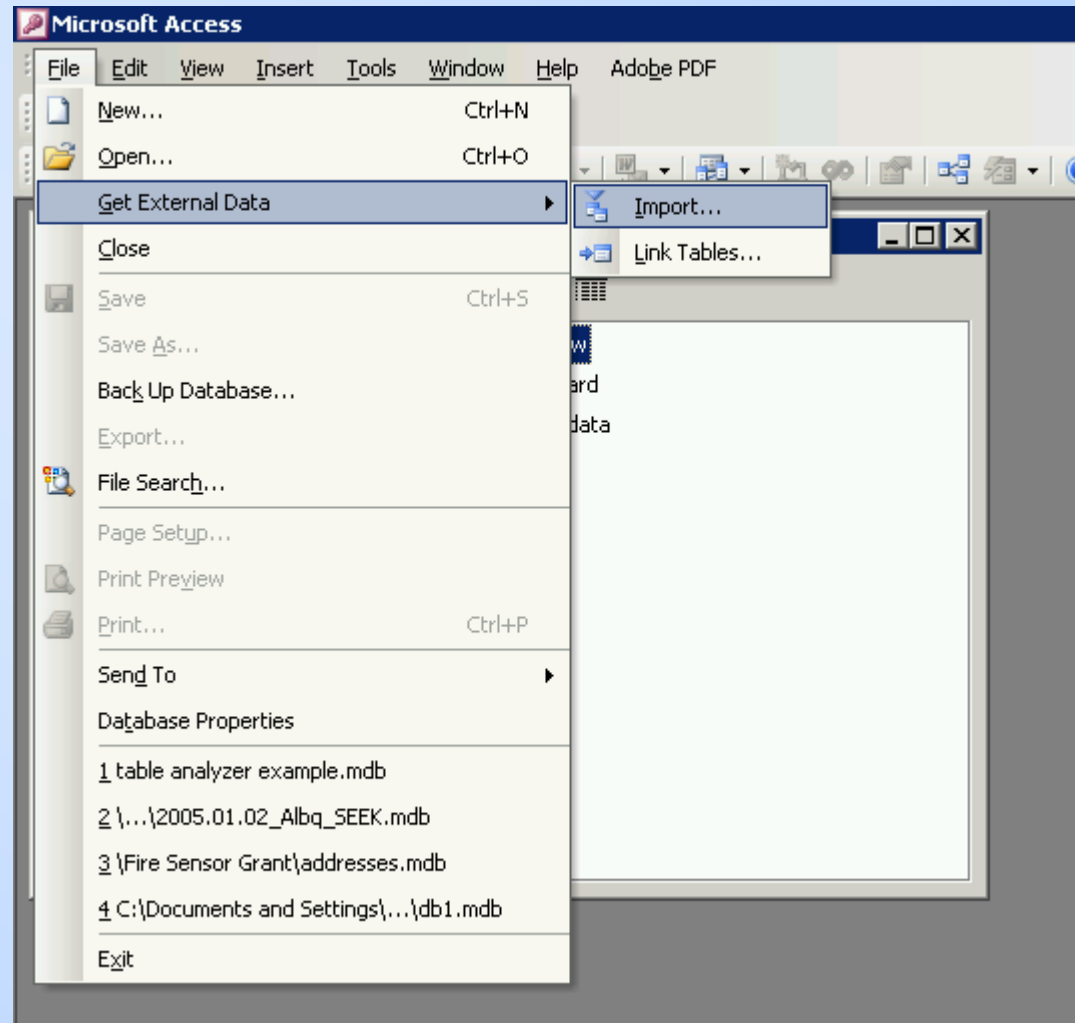
Species_id	Species_code
1	ERPU8
2	GUSA2
3	LEFE



# Normalizing Existing Data with MS Access – Step 1

Step 1:

Import data into  
Access as a single  
table.



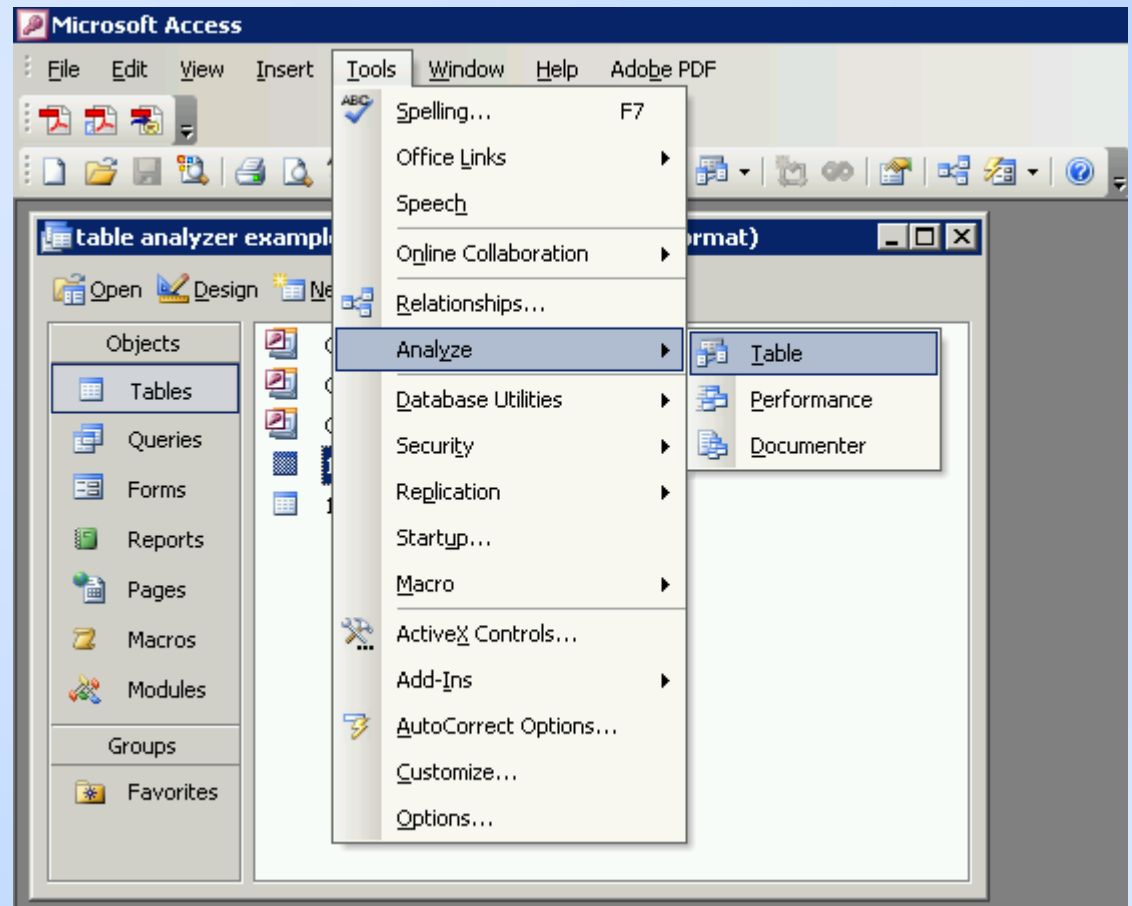


# Normalizing Existing Data with MS Access – Step 2

Step 2:

Run table analyzer  
tool.

(Can also launch right into it  
from **Import...** command)





# Normalizing Existing Data with MS Access – Step 2.1

Step 2.1:

a) Name tables

b) Move fields across tables

The screenshot shows the Microsoft Access Table Analyzer Wizard interface. The wizard is titled "Table Analyzer Wizard" and contains the following text:

Is the wizard grouping information correctly?  
If not, you can drag and drop fields to form groups that make sense.

What name do you want for each table?  
It's a good idea for the name to identify the kind of information the table contains.

The wizard displays a table relationship diagram with three tables:

- Location**: ID, SITE, WEB, PLOT, QD, Lookup to Observation, Lookup to Table3
- Observation**: DATE, COVER (primary key), COUNT, HEIGHT, COMMENTS
- Table3**: Generated Unique ID (primary key), SPECIES

Relationships are shown as lines connecting the tables:

- A 1-to-many relationship between Location and Observation, with the "1" at the Location end and the infinity symbol at the Observation end.
- A 1-to-many relationship between Location and Table3, with the "1" at the Location end and the infinity symbol at the Table3 end.

At the bottom of the wizard, there are four buttons: Cancel, < Back, Next >, and Finish.

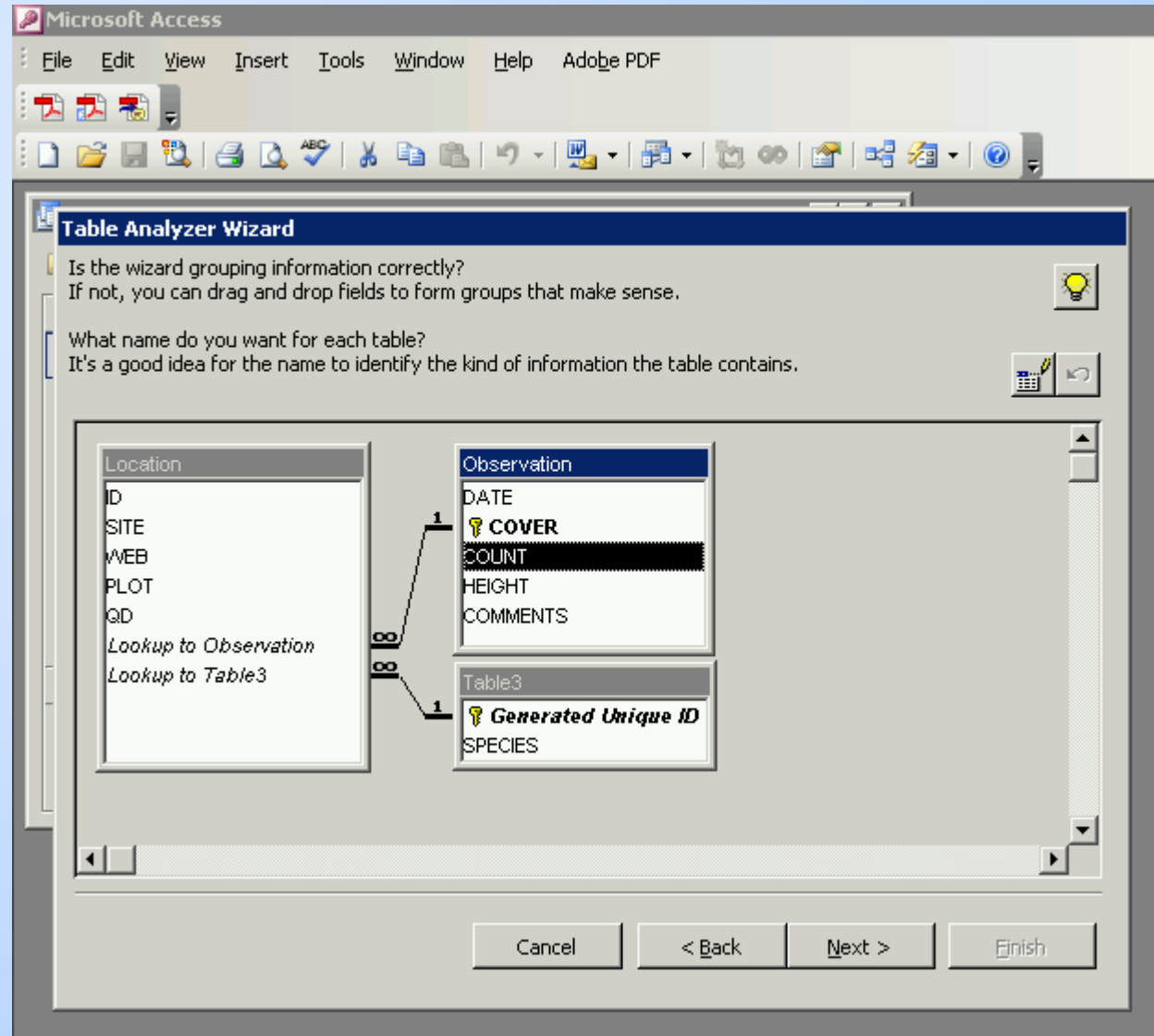


# Normalizing Existing Data with MS Access – Step 2.2

Step 2.2:

a) Name tables

b) Move fields across tables





# Normalizing Existing Data with MS Access – Step 2.3

Step 2.3:

Designate or create primary keys

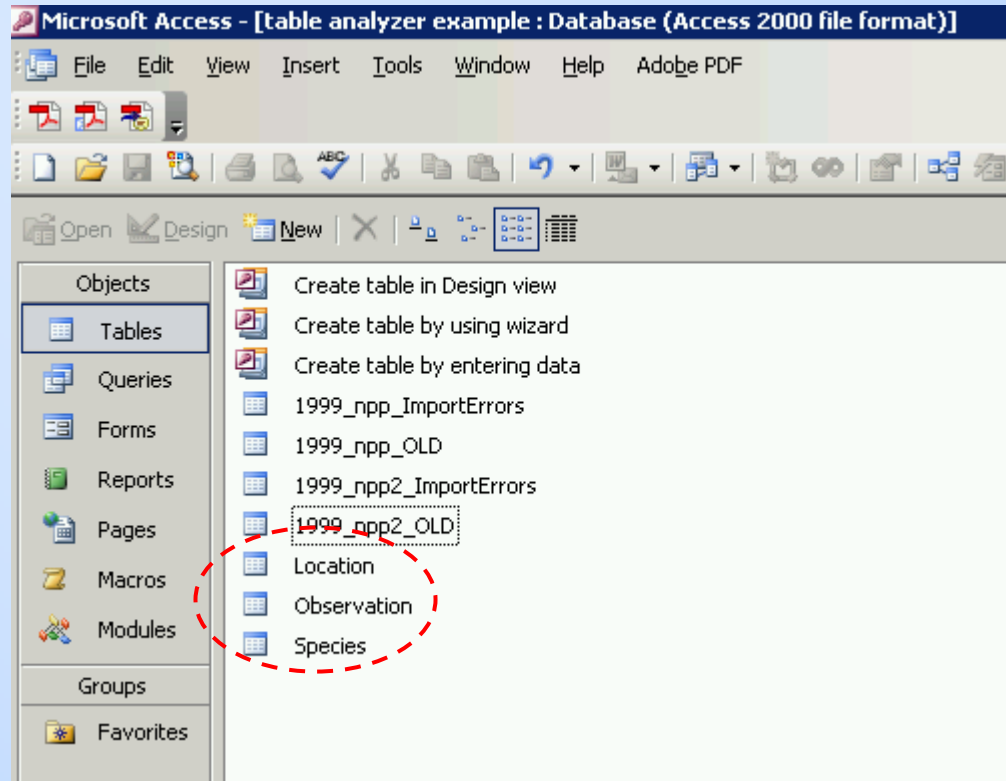
The screenshot shows the Microsoft Access Table Analyzer Wizard interface. The main question is "Do the bold fields uniquely identify each record in the proposed table?". Below this, it states: "The primary key field(s) must have a different value in every record of the proposed table. If no field has unique values, the wizard can add a Generated Unique ID field for you." In the top right corner, there are three buttons: a lightbulb icon, a key icon, and a plus sign icon. The key icon is circled in red. The main area displays two tables: "Location" and "Observation". The "Location" table has fields: ID, SITE, WEB, PLOT, QD, and two lookup fields: "Lookup to Observation" and "Lookup to Table3". The "Observation" table has fields: "Generated Unique ID" (bolded), DATE, COVER, COUNT, HEIGHT, and COMMENTS. Below the "Observation" table is another table, "Table3", with fields: "Generated Unique ID" (bolded) and SPECIES. A 1-to-many relationship is shown between the "ID" field in the "Location" table and the "Generated Unique ID" field in the "Observation" table. The relationship is indicated by a line with a "1" at the "Location" end and an "8" at the "Observation" end. At the bottom of the wizard, there are four buttons: "Cancel", "< Back", "Next >", and "Finish".



# Normalizing Existing Data with MS Access – Done

Done!

(Export tables)





# EXERCISES

I've copied NPP tables to your databases as **loc**, **obs**, and **spe**.  
MySQL documentation: <http://dev.mysql.com/doc/>

1. List all observations where height is less than 5.
2. List all observations at site **CM**.  
(Hint: best done with a *join*)
3. List all observations of species **BOGR2** at site **CM**.
4. Calculate average heights of all observations from (3).
5. Insert a new species into the **species** table.