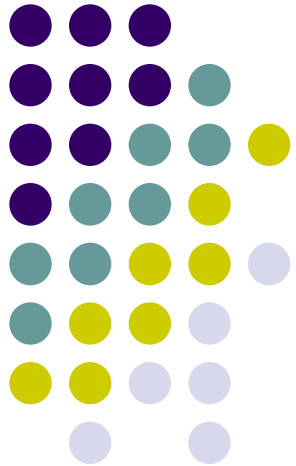
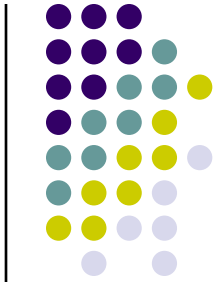
Databases and PHP

Storing and Retrieving information

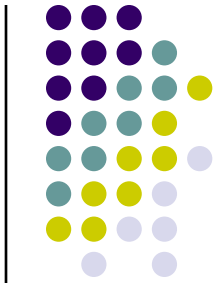


Database Basics



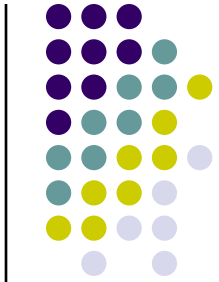
- A database is just information or data stored in a structured manner
- Database goal:
  - To organize some data in a manner that makes it easy to relate, store, and retrieve the data

Database Basics



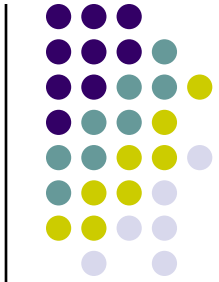
- Many different companies make databases tools that allow you to create, modify, and destroy databases:
  - Oracle (leading database)
  - MS SQL Server
  - MySQL (open source)
  - IBM DB2
  - MS Access
  - FileMaker Pro (cross platform)

Database Basics



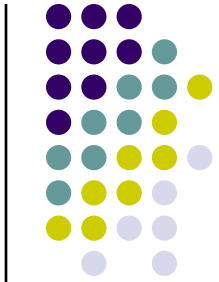
- What do we need to know about databases?
  - How to create a database
  - How to use and update a database

Database Basics



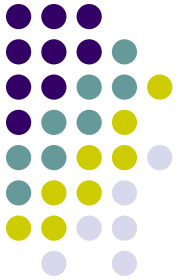
- Basic Database structure:
  - A database is a collection of tables
  - A table contains a set of records
  - All records have the same number of fields
  - Each field contains a particular piece of data

Database Basics



- Example: consider some random data that you want to store
  - Names (like Joe Smith)
  - Birth date (such as 18 Aug 1970)
  - Favorite color (eg, blue)
- Put our data in categories:
  - Name
  - Birth_Date
  - Fave_Color

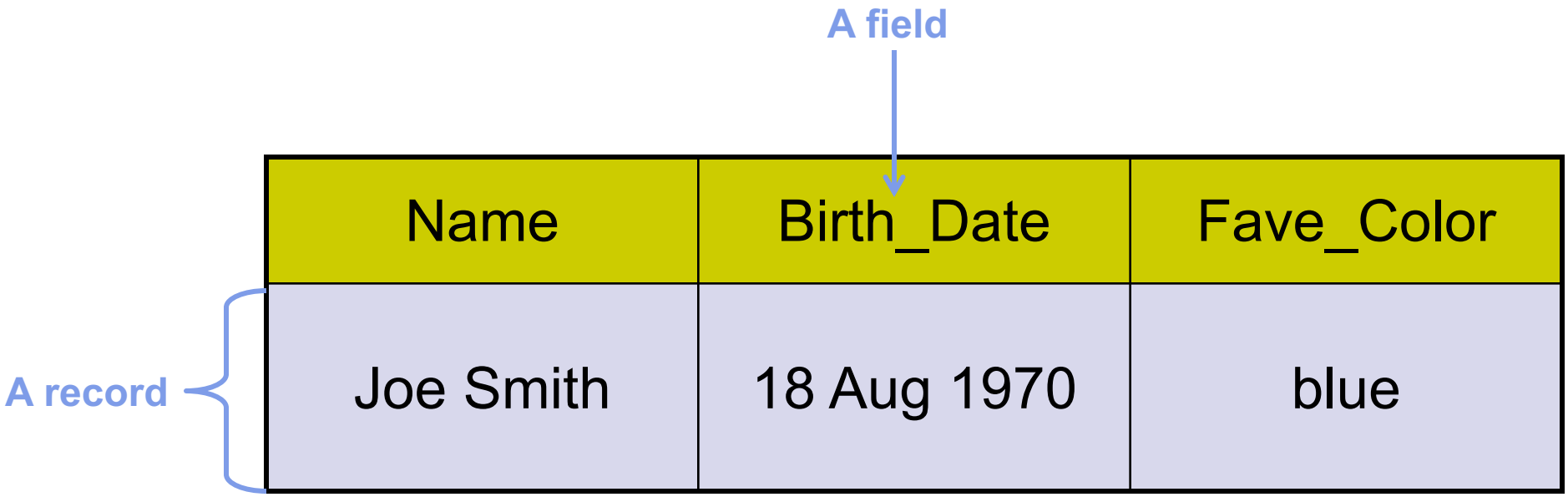
Database Basics



- Now organize our data in *tables*.
  - Each row will represent one person. Called a *record.*
  - Each column will represent one type of data. Called a *field*.

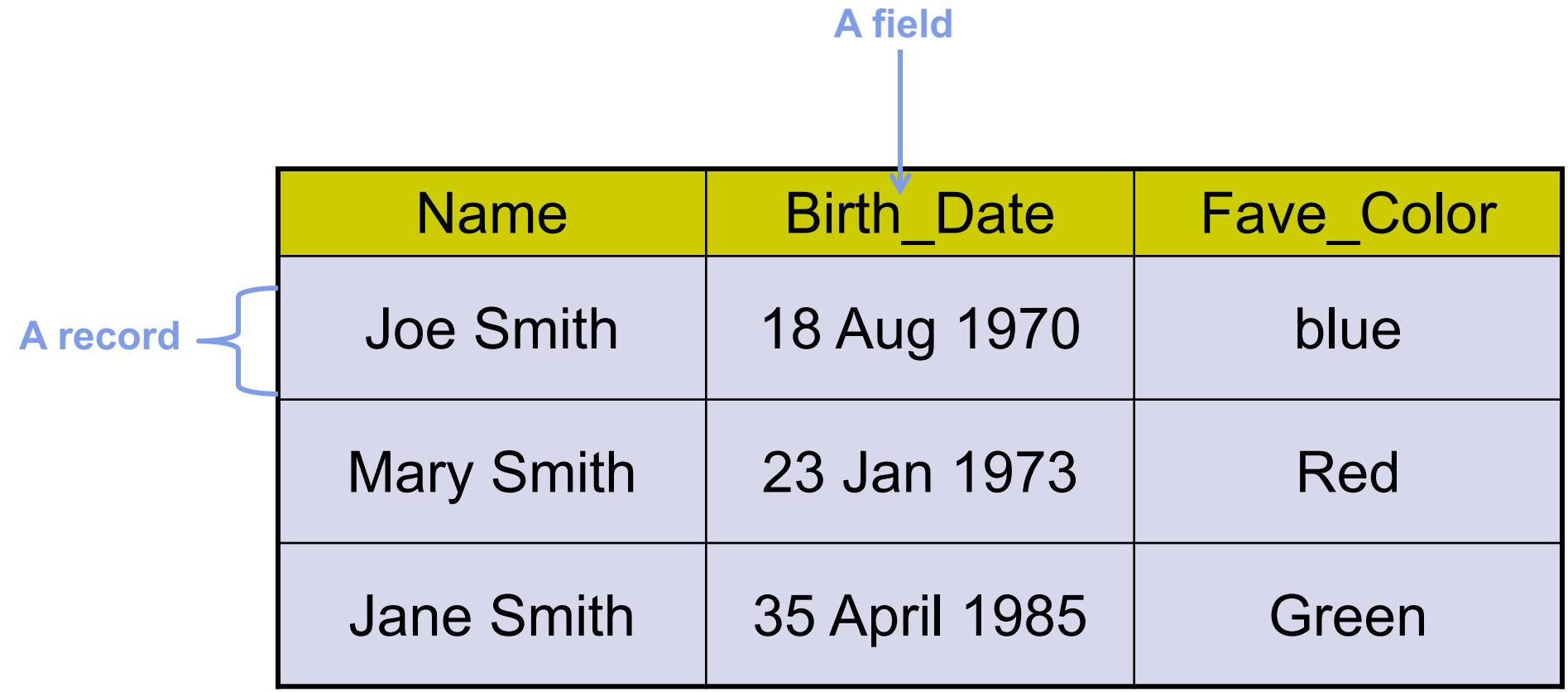
Database Basics

- Our example:



Database Basics

- Adding more records:



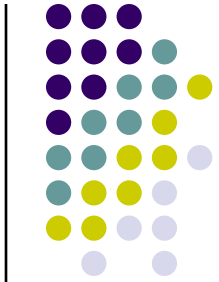
Database Basics

- Need to be able to *uniquely* identify records.
- Name field won't work; two people may have the same name.
- May have to add a unique field, like ID:

ID	Name	Birth_Date	Fave_Color
1	Joe Smith	18 Aug 1970	Blue
2	Mary Smith	23 Jan 1973	Red
3	Jane Smith	35 April 1985	Green
4	Joe Smith	9 Feb 1987	Purple

The unique field is called a ***key field***.

Database Basics



- A database may have many tables
- A *relational* database allows relationships to exist between tables
  - Relationships occur because the tables have fields in common (the *keys*).
- It is more efficient to create multiple tables and have relationships than it is to create one large table or to repeat data in tables

Database Basics

- Example: a product database.

Prod_id	Title	Descript	price
12557	Hat	Warm	7.50
12558	Jacket	Waterproof	32.50
12559	Shirt	Colorful	24.00
12560	Pants	Pleated	52.75
12561	Socks	Wool	14.99

Cust_ID	Name	Add
125	Mike	1212 Main St.
268	Jim	458 Bee Ave.
381	Nancy	751 1st St.

Order_ID	Customer	Prod_ordered	Quantity
1	125	12558	1
2	268	12558	2
3	125	12559	1

Red indicates a key field

Database Basics

- Example: a product database.

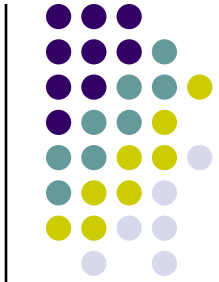
Prod_id	Title	Descript	price
12557	Hat	Warm	7.50
12558	Jacket	Waterproof	32.50
12559	Shirt	Colorful	24.00
12560	Pants	Pleated	52.75
12561	Socks	Wool	14.99

Cust_ID	Name	Add
125	Mike	1212 Main St.
268	Jim	458 Bee Ave.
381	Nancy	751 1st St.

Order_ID	Customer	Prod_ordered	Quantity
1	125	12558	1
2	268	12558	2
3	125	12559	1

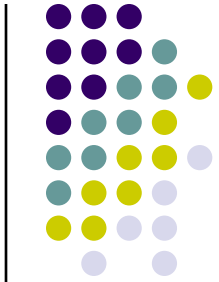
Red indicates a key field

Database Normalization



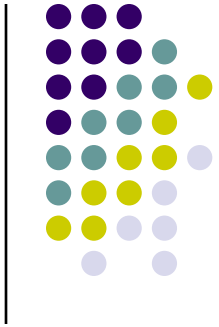
- Database normalization is a set of rules
  - These rules make you organize your DB such that tables are all related and flexible
  - Set of rules are called *normal forms*
  - If the first three sets of rules or normalization are followed, the database is said to be in *third normal form*

Database Normalization



- Flat table
  - No organization of data
  - No multiple tables
  - All data in one giant table
  - Like a spreadsheet with many columns for data

Database Normalization

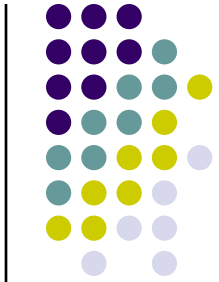


- Flat table example: data repeated, no relationships

Student Name	CourseID1	Course Description1	Course Instructor1	CourseID2	Course Description 2	Course Instructor2	Etc.
George	304-212	Stuff	Albert	319-291	Junk	Susan	
Julie	319-291	Junk	Susan	304-245	Stars	Albert	
Sam	304-212	Stuff	Albert	319-291	Junk	Susan	
Jessica	304-245	Stars	Albert	304-212	Stuff	Albert	

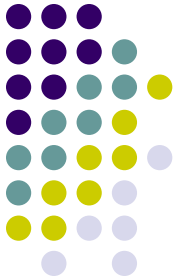
Note all the duplication!

Database Normalization



- **First Normal Form**
  - Eliminate repeating information
  - Create separate tables for related data
- Example table has two main topics:
  - Students
  - Courses
- First normal form example would create two tables
  - Students (*students*)
  - Students + courses (*students_courses*)

Database Normalization



- 1-to-many relationship: one student to many courses
- Number of courses a student may take is now not limited to the number of columns in the table.

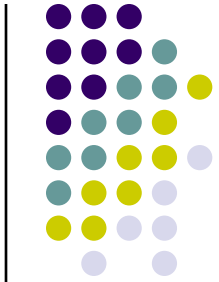
students table

StudentID	StudentName
12123	George
98987	Julie

StudentID	CourseID	Course Description	Course Instructor
12123	304-212	Stuff	Albert
12123	319-291	Junk	Susan
98987	319-291	Junk	Susan
98987	304-245	Stars	Albert

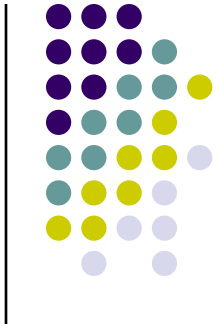
students_courses table

Database Normalization



- **Second Normal Form**
  - No non-key attributes depend on a portion of the primary key.
- ie, if fields in your table are not entirely related to a primary key, they must go.
- In our example: students should not be in the courses table.
- Now have three tables:
  - Students table (as before)
  - Courses table (course ID, description, instructor)
  - Student_Courses table (student id, course id)

Database Normalization



- 1-to-many relationship: one student to many courses

students table

StudentID	StudentName
12123	George
98987	Julie

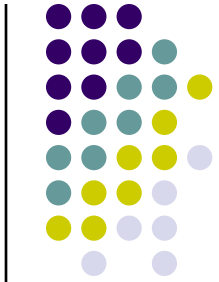
CourseID	Course Description	Course Instructor
304-212	Stuff	Albert
319-291	Junk	Susan
319-291	Junk	Susan
304-245	Stars	Albert

courses table

StudentID	CourseID
12123	304-212
12123	319-291
98987	319-291
98987	304-245

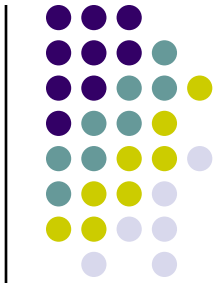
students_courses table

Database Normalization



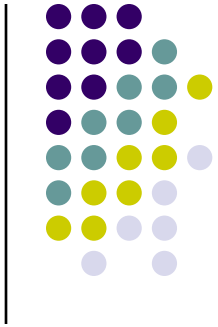
- **Third Normal Form**
  - No attributes depend on other non-key attributes.
- Ie, see if more fields exist that can be broken down further and that aren't dependent on a key.
- Think about removing data
- In our example: instructors
  - Instructors can teach more than one class
  - However, the *courseInstructor* field in the courses table is not a key of any sort.
  - So can break this field out into its own table
  - In reality, would have more info about instructors (instructorID, etc.) that would go into this new table.

Database Normalization



- **Third Normal Form**
  - No attributes depend on other non-key attributes.
- 1-to-many relationship: one student to many courses
- 1-to-many relationship: one instructor to many courses.
- Now have 4 tables:
  - Students table (as before)
  - Courses table (course ID, description, instructor)
  - Student_Courses table (student id, course id)
  - Instructors table (instructor ID, name)

Database Normalization



instructors table

InstructorID	InstructorName
56564	Albert
76765	Susan

students table

StudentID	StudentName
12123	George
98987	Julie

CourseID	Course Description	Instructor ID
304-212	**Stuff**	56564
319-291	**Junk**	76765
304-245	**Stars**	56564

courses table

StudentID	CourseID
12123	**304-212**
12123	**319-291**
98987	**319-291**
98987	**304-245**

students_courses table