Announcements

- Quiz will cover chapter 16 in *Fluency*
  - Nothing in QuickStart
- Read Chapter 17 for Wednesday
- Project 3
  - 3A due Friday before 11pm
  - 3B due Monday, March 17 before 11pm

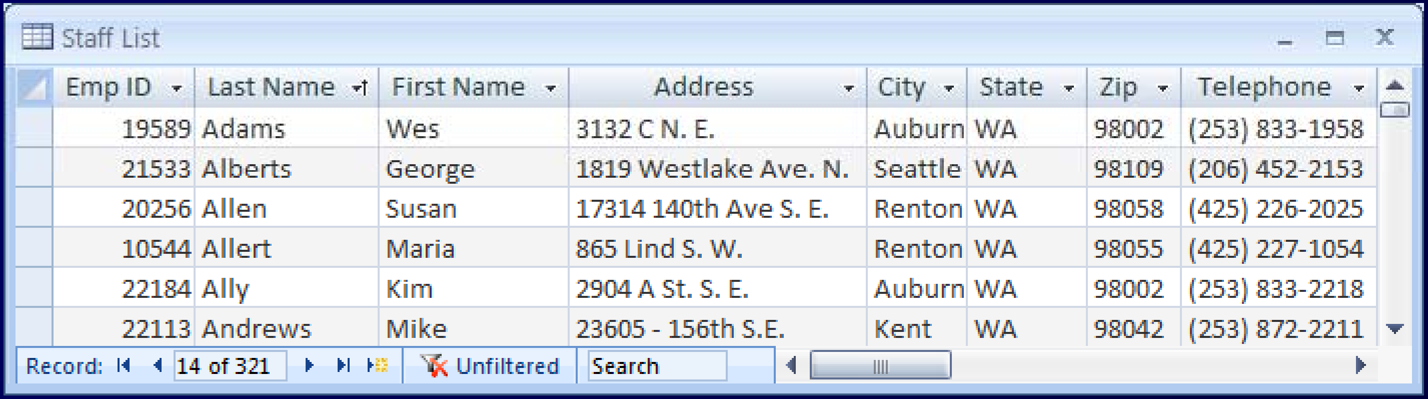
A Table with a View (continued)

Primary keys, normalization, and SQL

Video

- Primary Keys

Fields (Attributes) and Primary Keys



Emp ID	Last Name	First Name	Address	City	State	Zip	Telephone
19589	Adams	Wes	3132 C N. E.	Auburn	WA	98002	(253) 833-1958
21533	Alberts	George	1819 Westlake Ave. N.	Seattle	WA	98109	(206) 452-2153
20256	Allen	Susan	17314 140th Ave S. E.	Renton	WA	98058	(425) 226-2025
10544	Allert	Maria	865 Lind S. W.	Renton	WA	98055	(425) 227-1054
22184	Ally	Kim	2904 A St. S. E.	Auburn	WA	98002	(253) 833-2218
22113	Andrews	Mike	23605 - 156th S.E.	Kent	WA	98042	(253) 872-2211

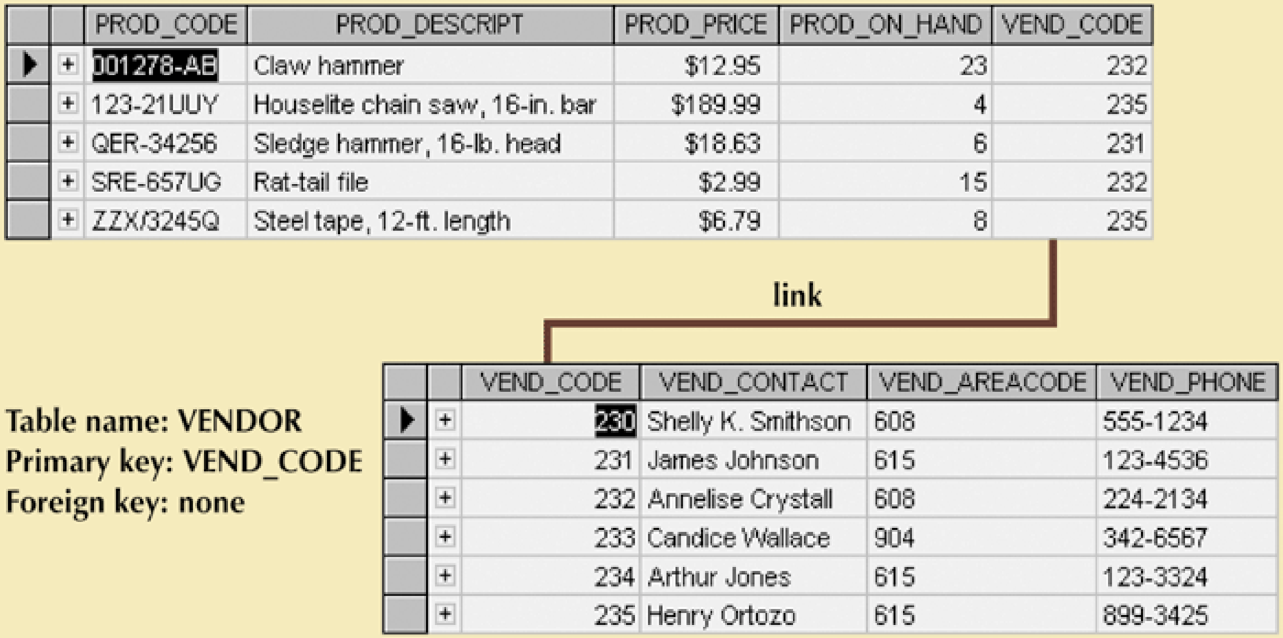
- Primary Key (PK)
  - Field or attribute that uniquely identifies each entity (row)

Keys – Primary & Foreign

FIGURE 3.2 An example of a simple relational database

Table name: PRODUCT
Primary key: PROD_CODE
Foreign key: VEND_CODE

Database name: Ch03_SaleCo

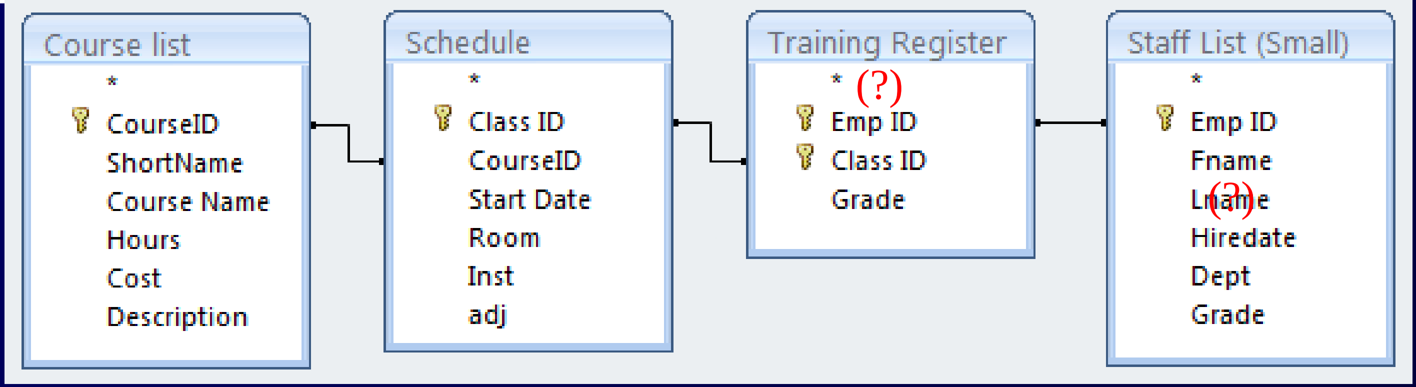


PROD_CODE	PROD_DESCRIPT	PROD_PRICE	PROD_ON_HAND	VEND_CODE
001278-AB	Claw hammer	$12.95	23	232
123-21UUY	Houselite chain saw, 16-in. bar	$189.99	4	235
QER-34256	Sledge hammer, 16-lb. head	$18.63	6	231
SRE-657UG	Rat-tail file	$2.99	15	232
ZZX/3245Q	Steel tape, 12-ft. length	$6.79	8	235

Table name: VENDOR
Primary key: VEND_CODE
Foreign key: none

VEND_CODE	VEND_CONTACT	VEND_AREACODE	VEND_PHONE
230	Shelly K. Smithson	608	555-1234
231	James Johnson	615	123-4536
232	Annelise Crystall	608	224-2134
233	Candice Wallace	904	342-6567
234	Arthur Jones	615	123-3324
235	Henry Ortozo	615	899-3425

Primary/Foreign Key



Course list
- *
- CourseID
- ShortName
- Course Name
- Hours
- Cost
- Description

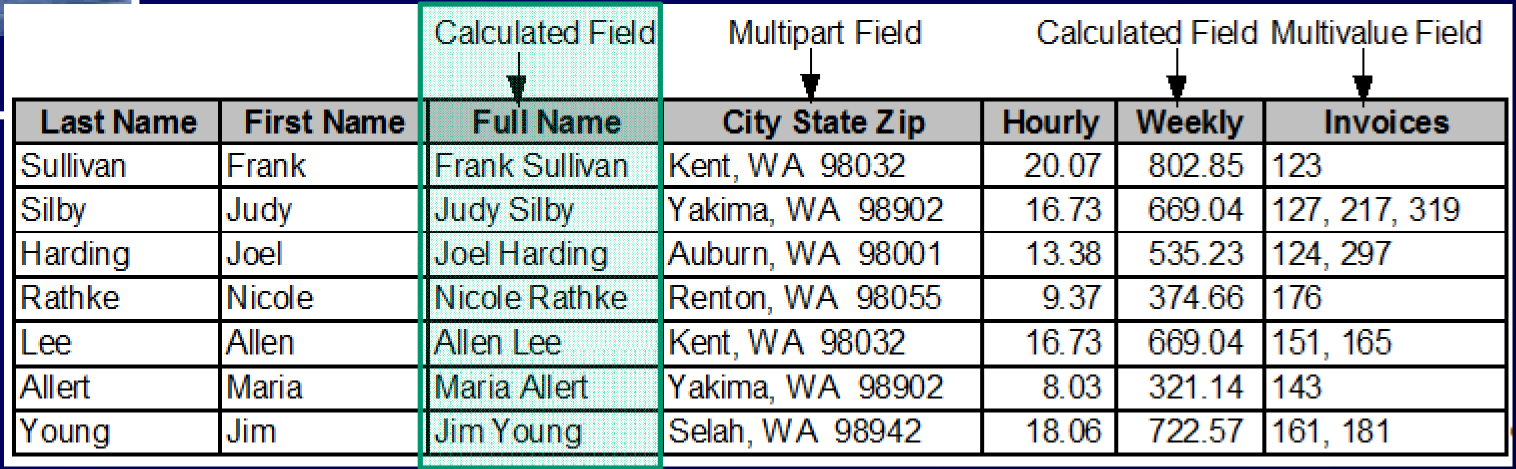
Schedule
- *
- Class ID
- CourseID
- Start Date
- Room
- Inst
- adj

Training Register
- * (?)
- Emp ID
- Class ID
- Grade

Staff List (Small)
- *
- Emp ID
- Fname
- Lname (?)
- Hiredate
- Dept
- Grade

- Controlled redundancy:
  - Stores relationship between tables
  - Database tables share common attributes only to enable the tables to be linked
  - True redundancy exists only when there is unnecessary duplication of attribute values

Problem Fields (Don’ts)



Calculated Field

Multipart Field

Calculated Field

Multivalue Field

Last Name	First Name	Full Name	City State Zip	Hourly	Weekly	Invoices
Sullivan	Frank	Frank Sullivan	Kent, WA 98032	20.07	802.85	123
Silby	Judy	Judy Silby	Yakima, WA 98902	16.73	669.04	127, 217, 319
Harding	Joel	Joel Harding	Auburn, WA 98001	13.38	535.23	124, 297
Rathke	Nicole	Nicole Rathke	Renton, WA 98055	9.37	374.66	176
Lee	Allen	Allen Lee	Kent, WA 98032	16.73	669.04	151, 165
Allert	Maria	Maria Allert	Yakima, WA 98902	8.03	321.14	143
Young	Jim	Jim Young	Selah, WA 98942	18.06	722.57	161, 181

- Calculated field – can be computed by mathematical calculation or text concatenation
  - Waste of storage space (redundant),
  - No assurance the calculated value is updated when the user changes the input field(s)
- Multipart field – contains that should be two or more fields
  - Extra work when you want to analyze your data
- Multivalue field – multiple correct entries for the field
  - Create a separate subset table with each value in its own record.
- Derived field – contents of one or more fields absolutely predicts the contents of another
  - Should be dropped from the table

Video

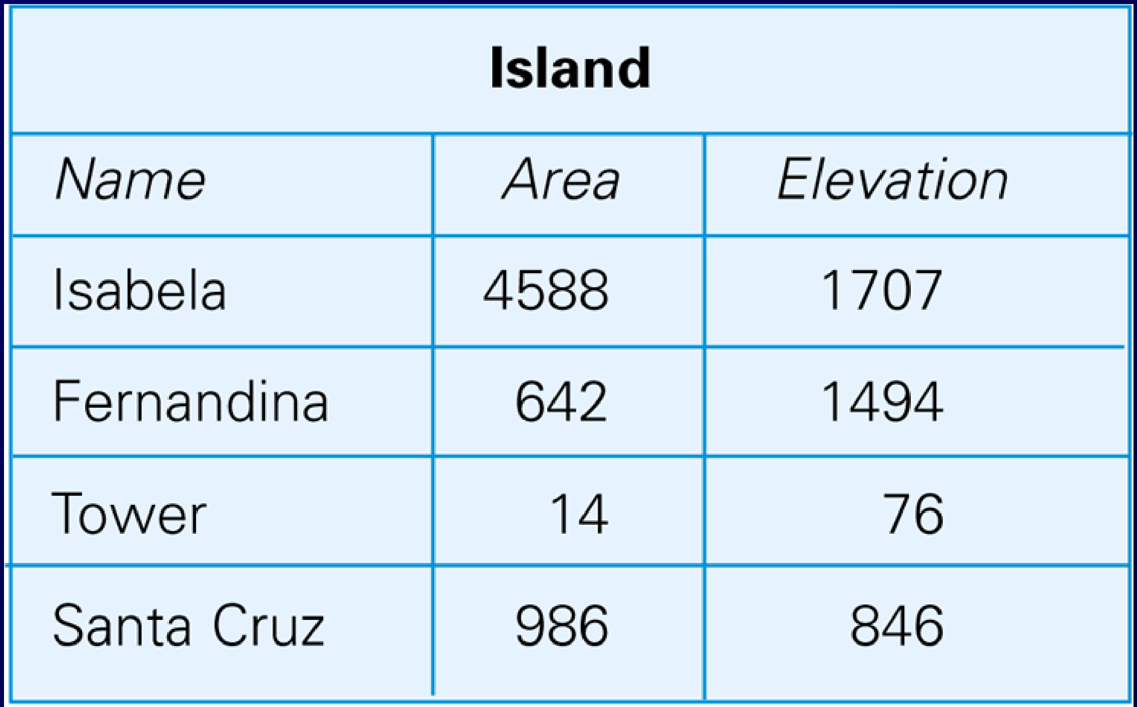
- Redundancy and Normalization

Entities

- Entity
  - Anything that can be identified by a fixed number of its characteristics (*attributes*)
- Attributes have
  - Names—field name, attribute, or column name
  - Values—the data stored in the table

Entities

- An entity defines a table
  - Name of the entity is the name of the table
  - Each attribute of that entity
    - The column heading is the attribute name



Island		
Name	*Area*	*Elevation*
Isabela	4588	1707
Fernandina	642	1494
Tower	14	76
Santa Cruz	986	846

Figure 16.4 A table instance for the island entity.

Properties of Entities

- A relational database table can be empty
- Instances Are Unordered
  - ∗ Order of the rows and columns does not matter in databases
  - ∗ Freedom to move the data is limited to exchanging entire rows or exchanging entire columns

Properties of Entities cont'd)

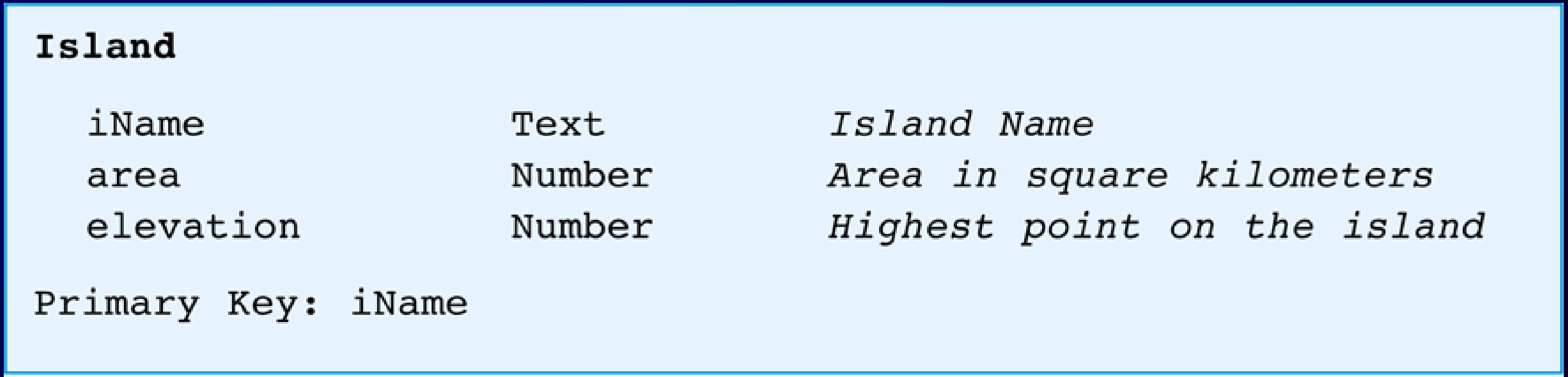
- Uniqueness
  - No two rows can be the same
  - Two rows can have the same value for some attributes, just not all attributes

Properties Of Entities (cont'd)

- Atomic Data
  - ∗ Not decomposable into any smaller parts
    - Separate fields for street, city, state, postal code
  - ∗ "Only atomic data" rule relaxed for certain types of data
    - Dates, times, currency

Database schemes

- Database schema – way to define a table
  - Collection of table definitions that gives the name of the table, lists the attributes and their data types, and identifies the primary key



Island

iName	Text	*Island Name*
area	Number	*Area in square kilometers*
elevation	Number	*Highest point on the island*

Primary Key: iName

Figure 16.5 Database table definition for an Island table.

Database Tables Recap

- Tables in databases have a structure that is specified by metadata
- The structure is separate from its content
- A table structures a set of entities
  - Things that we can tell apart by their attributes
- The entities of the table are represented as rows
  - Rows and columns are unordered
- Tables and fields should have names that describe their contents
  - Fields must be atomic (indivisible)
  - One of more attributes define the primary key

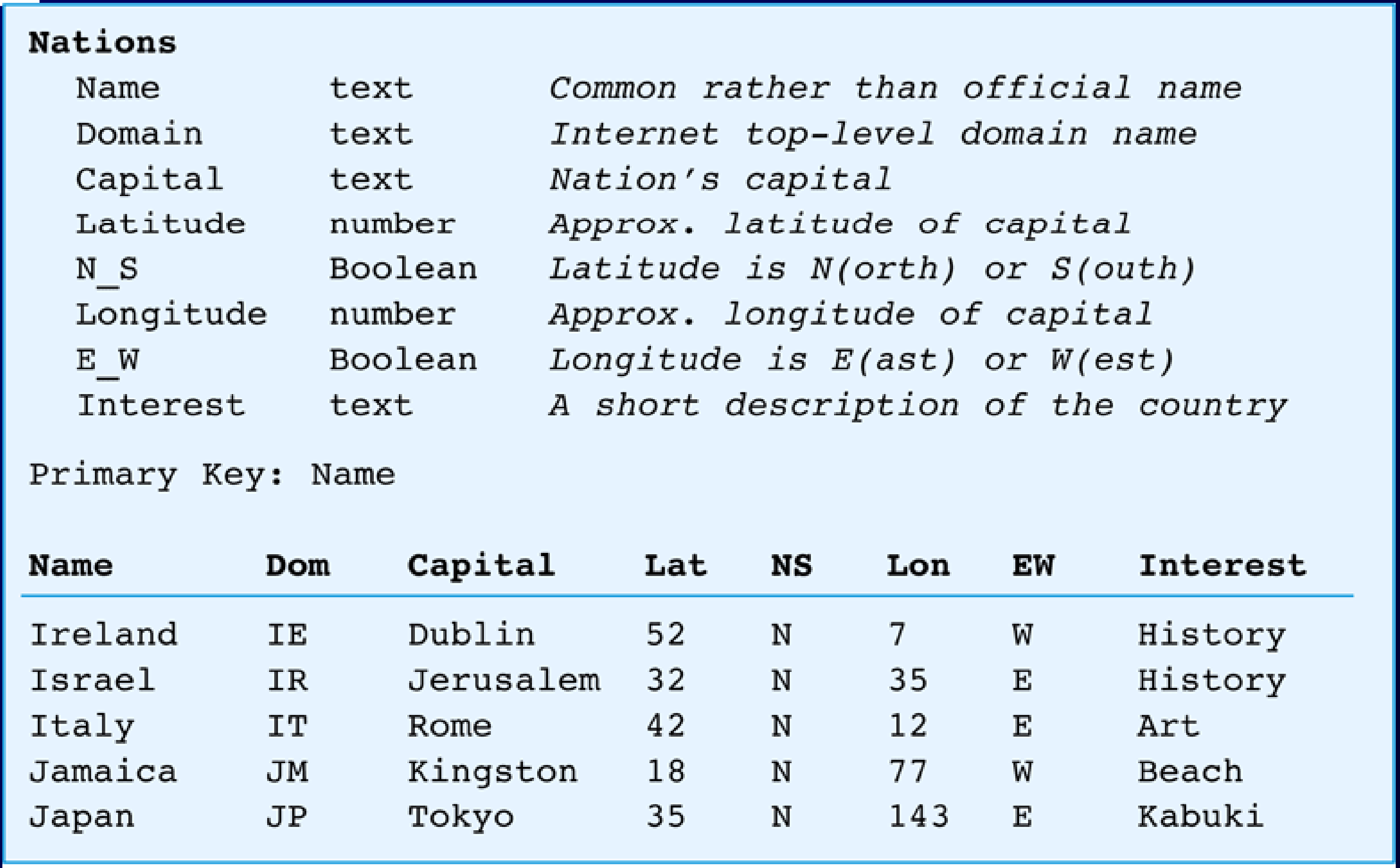
TABLE OPERATIONS

Operations on Tables

- A database is a collection of tables
- Main use of database is to look up information
  - Users specify what they want to know and the database software finds it
- We can perform operations on tables to produce tables
- The questions we ask of a database are answered with a whole new table, or view

Operations on Tables

- Five fundamental operations can be performed on tables:
  - Select
  - Project
  - Union
  - Difference
  - Product
- Join



Nations

Name	text	*Common rather than official name*
Domain	text	*Internet top-level domain name*
Capital	text	*Nation's capital*
Latitude	number	*Approx. latitude of capital*
N_S	Boolean	*Latitude is N(orth) or S(outh)*
Longitude	number	*Approx. longitude of capital*
E_W	Boolean	*Longitude is E(ast) or W(est)*
Interest	text	*A short description of the country*

Primary Key: Name

Name	**Dom**	**Capital**	**Lat**	**NS**	**Lon**	**EW**	**Interest**
Ireland	IE	Dublin	52	N	7	W	History
Israel	IR	Jerusalem	32	N	35	E	History
Italy	IT	Rome	42	N	12	E	Art
Jamaica	JM	Kingston	18	N	77	W	Beach
Japan	JP	Tokyo	35	N	143	E	Kabuki

Figure 16.6 The Nations table definition and sample entries.

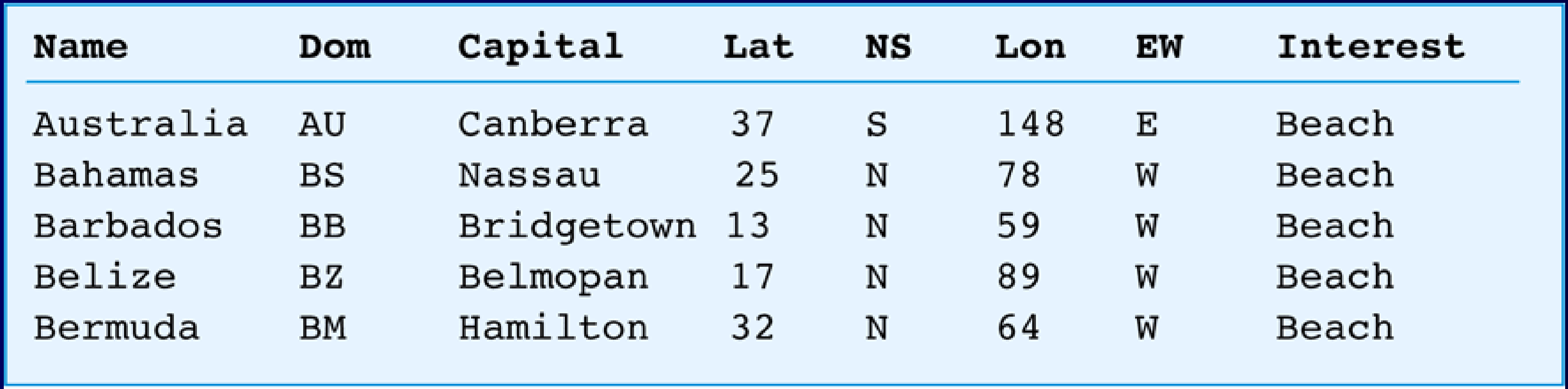
Select Operation

- Takes rows from one table to create a new table
  - ∗ Specify the table from which rows are to be taken, and the *test* for selection

    Syntax: **SELECT** *Test* **FROM** *Table*

  - ∗ Test is applied to each rows of the table to determine if it should be included in result table
  - ∗ Test uses attribute names, constants, and relational operators
  - ∗ If the test is true for a given row, the row is included in the result table; otherwise it is ignored

    SELECT Interest='Beach' **FROM** *Nations*



Name	Dom	Capital	Lat	NS	Lon	EW	Interest
Australia	AU	Canberra	37	S	148	E	Beach
Bahamas	BS	Nassau	25	N	78	W	Beach
Barbados	BB	Bridgetown	13	N	59	W	Beach
Belize	BZ	Belmopan	17	N	89	W	Beach
Bermuda	BM	Hamilton	32	N	64	W	Beach

Figure 16.7 Part of the table created by selecting countries with a Test for Interest equal to Beach.

Join Operation

- Combines two tables, like the Product Operation, but doesn't necessarily produce all pairings
  - ∗ If the two tables each have fields with a common data type, the new table combines only the rows from the given tables that match on the fields
  - ∗ Syntax: *Table1* ⋈ *Table2* **On** *Match*

Animation

- A natural join

Physical and Logical Database

TABLES AND VIEWS

Structure of a Database

- Physical database and logical database
  - ∗ Physical database is the files, records in any order, no logical organization other than tables
  - ∗ Logical database is a view of database that shows only the rows and fields needed by the users
    - Solves Information Overload:
      - – Show users only what they need to see

Physical vs. Logical

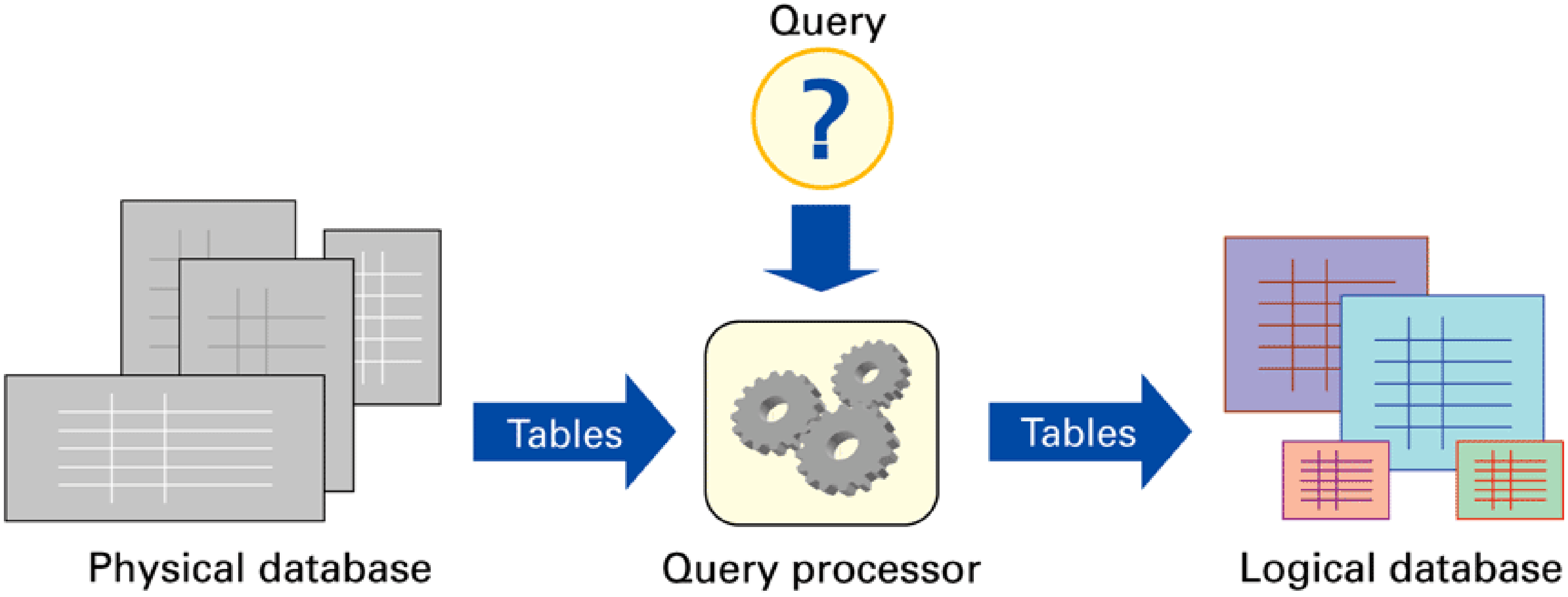


Figure 16.15 Structure of a database system. The physical database is the permanent repository of the data; the logical database, or view of the database, is the form of the database the users see. The transformation is implemented by the query processor, and is based on queries that define the logical database tables from the physical database tables.

Physical Database

- Designed by database administrators
  - ∗ Fast to access
  - ∗ No redundancy/duplicating information
    - Multiple data can lead to inconsistent data
  - ∗ Backup copies in case of accidental data deletion or disk crash

Logical Database

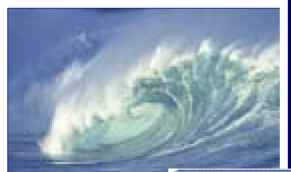
- Creating specialized views of the data for different users' needs
  - ∗ Creating a new “result set” from the current data each time
    - Fresh
    - Accurate

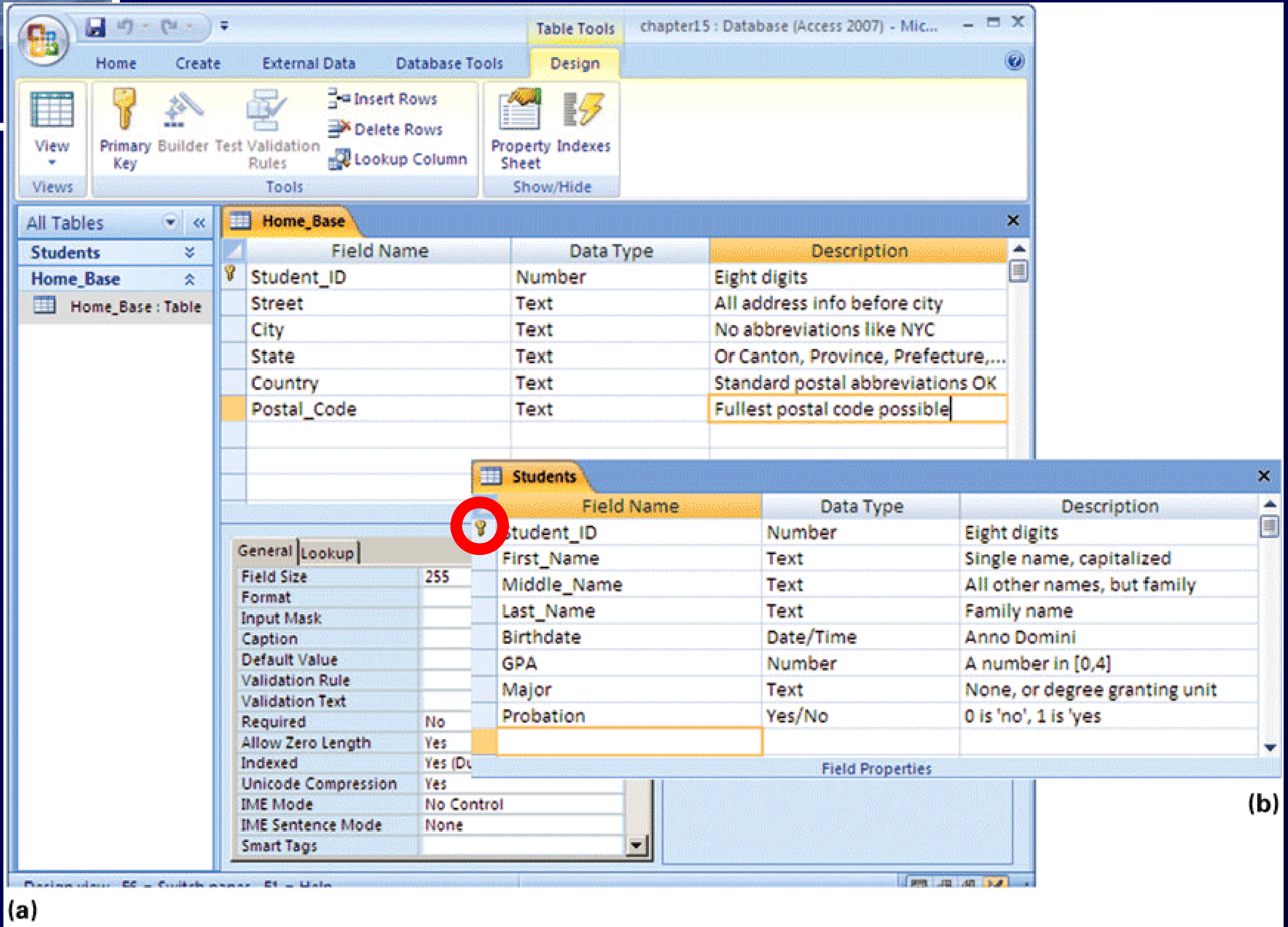
Queries

- A query is a specification using the five operations and Join that create a view from other tables
- SQL (Structured Query Language)
  - ∗ Standard database language to write queries

Defining Physical Tables

- Database schemes (schema)
  - Metadata specification that describes the database design





Home_Base

Field Name	Data Type	Description
Student_ID	Number	Eight digits
Street	Text	All address info before city
City	Text	No abbreviations like NYC
State	Text	Or Canton, Province, Prefecture,...
Country	Text	Standard postal abbreviations OK
Postal_Code	Text	Fullest postal code possible

Field Size	255
Format	
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	Yes
Indexed	Yes (Du
Unicode Compression	Yes
IME Mode	No Control
IME Sentence Mode	None
Smart Tags	

Students

Field Name	Data Type	Description
tudent_ID	Number	Eight digits
First_Name	Text	Single name, capitalized
Middle_Name	Text	All other names, but family
Last_Name	Text	Family name
Birthdate	Date/Time	Anno Domini
GPA	Number	A number in [0,4]
Major	Text	None, or degree granting unit
Probation	Yes/No	0 is 'no', 1 is 'yes

Figure 16.16 Table declarations from Microsoft Access 2007: (a) `Home_Base` table declaration shown in the design view; and (b) students table declaration. Notice that the key is specified by the tiny key next to `Student_ID` in the first column.

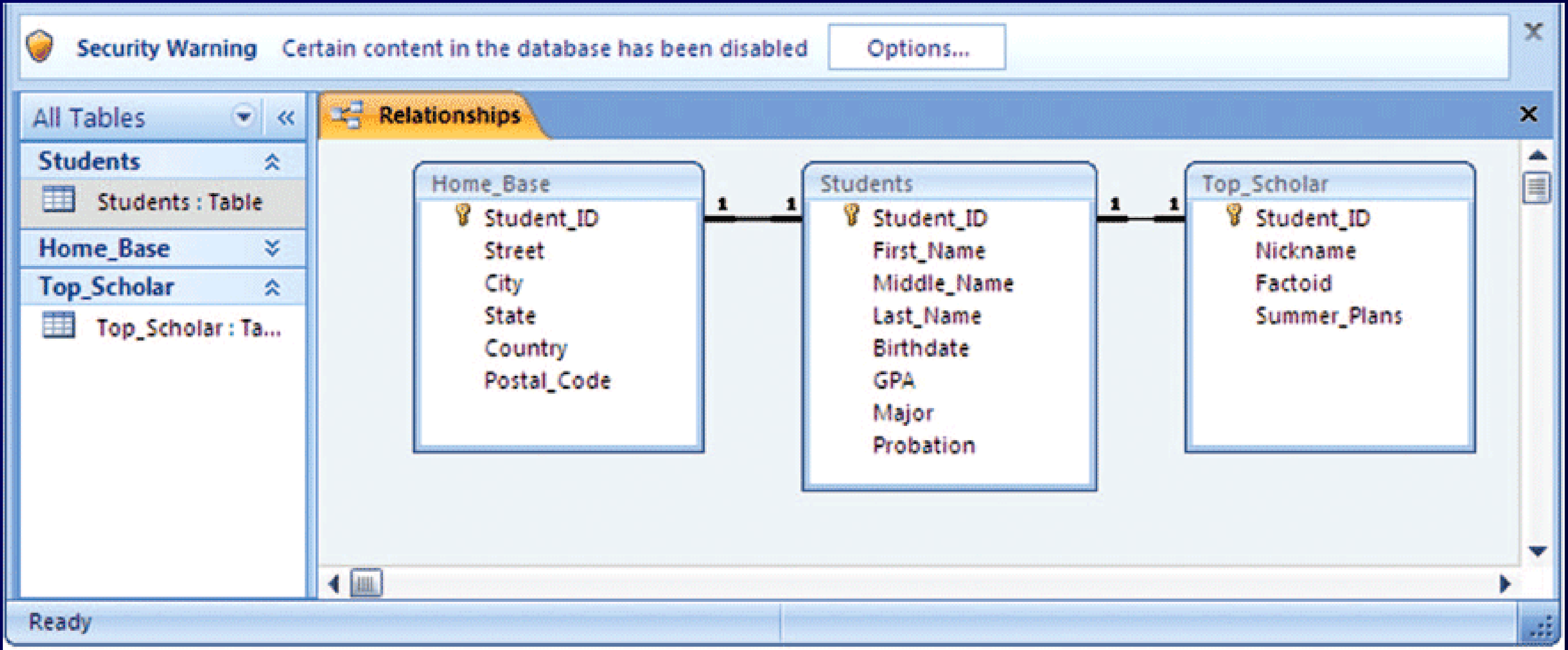
Connecting Database Tables by Relationships

- Student and Home_Base tables
  - The tables can have different security access restrictions based on their data
    - Other units can access Home_Base data without having access to more sensitive data in Student
  - Separate tables but not independent
    - Student_ID connects (establishes a relationship) the two tables
      - Primary key in one, foreign key in the other

The Idea of Relationship

- A **relationship** is a correspondence between rows of one table and the rows of another table
  - Because the key Student_ID is used in each table, can not only find the address for each student (*Lives_At*), but can also find the student for each address (*Home_Of*)
- Relationship examples

Relationships in Practice



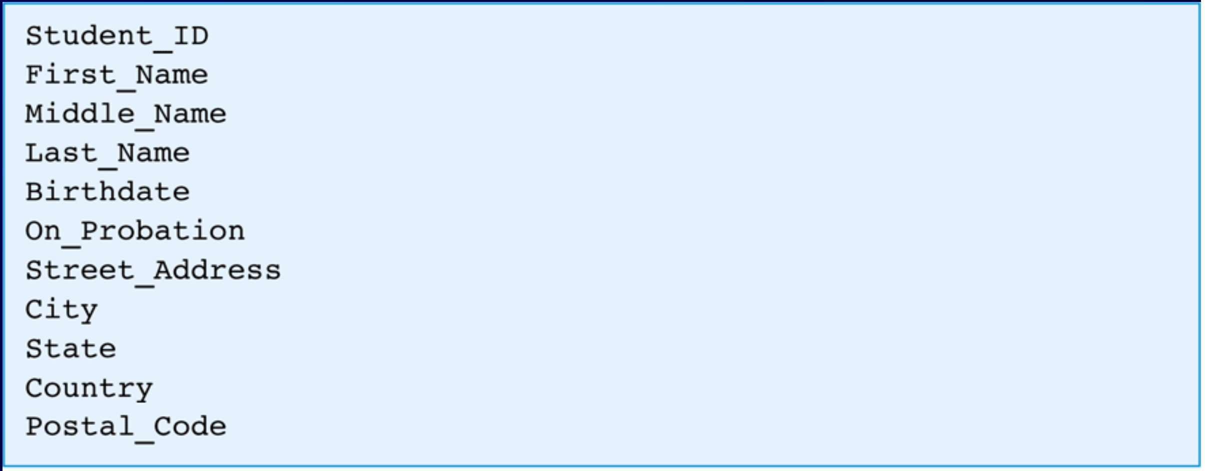
Security Warning Certain content in the database has been disabled

Figure 16.17 The *Relationships* window from the Microsoft Access database system; the 1-to-1 *Lives_At* and *Home_Of* relationships are shown between Home_Base and Students.

Defining Logical Tables

- Constructing a View Using Join
  - ∗ Match on the common field of Student_ID

```
Master_List = Student JOIN Home_Base
  On Student.Student_ID = Home_Base.Student_ID
```

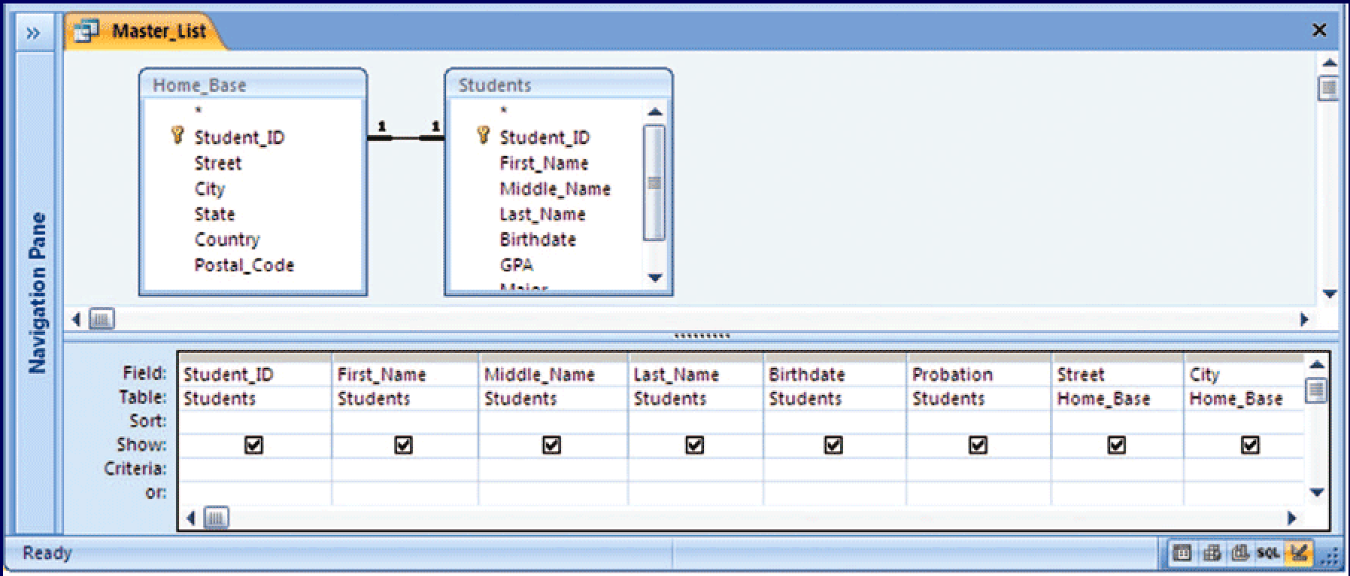


```
Student_ID
First_Name
Middle_Name
Last_Name
Birthdate
On_Probation
Street_Address
City
State
Country
Postal_Code
```

Figure 16.18 Attributes of the Master_List table. Being created from Student and Home_Base allows Master_List to inherit its data types and key (Student_ID) from the component tables.

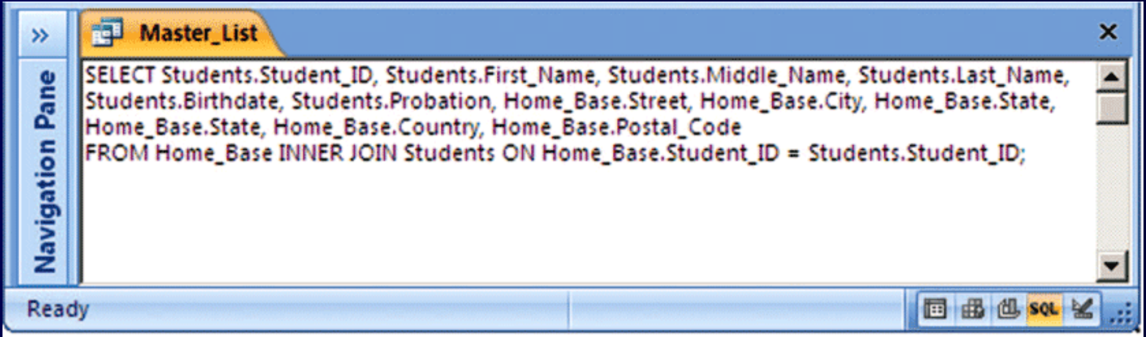
Practical Construction Using QBE

- Query By Example
  - Given a template of a table we fill in what we want in the fields



Field:	Student_ID	First_Name	Middle_Name	Last_Name	Birthdate	Probation	Street	City
Table:	Students	Students	Students	Students	Students	Students	Home_Base	Home_Base
Sort:								
Show:	☑	☑	☑	☑	☑	☑	☑	☑
Criteria:								
or:								

Figure 16.19 The Query By Example definition of the `Master_List` table from MS Access.

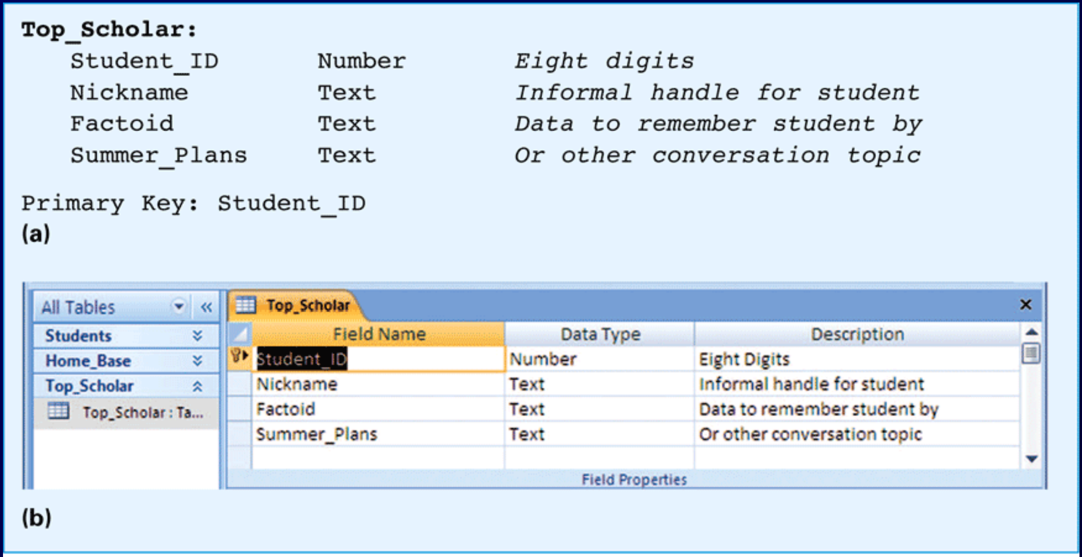


```
SELECT Students.Student_ID, Students.First_Name, Students.Middle_Name, Students.Last_Name,
Students.Birthdate, Students.Probation, Home_Base.Street, Home_Base.City, Home_Base.State,
Home_Base.State, Home_Base.Country, Home_Base.Postal_Code
FROM Home_Base INNER JOIN Students ON Home_Base.Student_ID = Students.Student_ID;
```

Figure 16.20 SQL query created from the Query By Example data in Figure 16.19.

The Dean's View

- Storing the Dean's Data
  - Top_Scholar is information of interest only to the dean



```
Top_Scholar:
    Student_ID      Number       Eight digits
    Nickname        Text         Informal handle for student
    Factoid         Text         Data to remember student by
    Summer_Plans    Text         Or other conversation topic

Primary Key: Student_ID
```

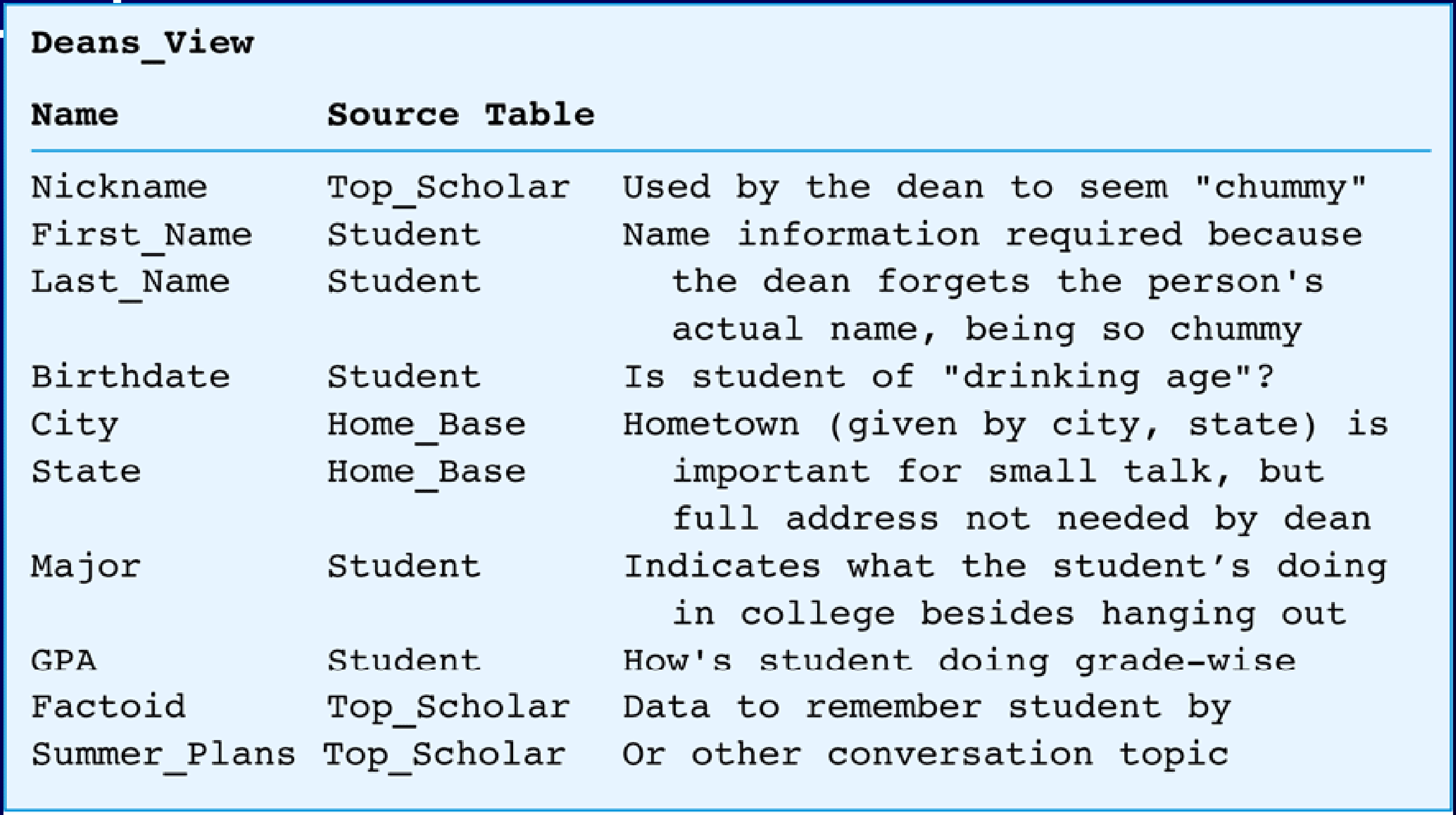
(a)

Field Name	Data Type	Description
Student_ID	Number	Eight Digits
Nickname	Text	Informal handle for student
Factoid	Text	Data to remember student by
Summer_Plans	Text	Or other conversation topic

(b)

Figure 16.21 The `Top_Scholar` definition: (a) informal form, (b) in MS Access.

Creating a Dean's View



Deans_View

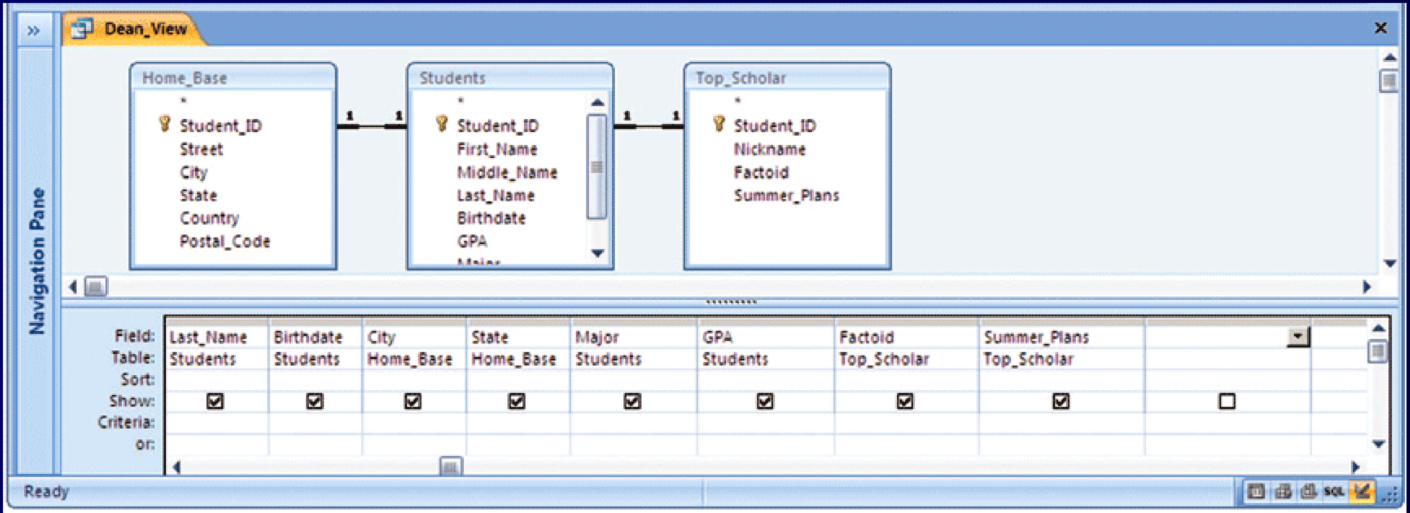
Name	**Source Table**	
Nickname	Top_Scholar	Used by the dean to seem "chummy"
First_Name	Student	Name information required because the dean forgets the person's actual name, being so chummy
Last_Name	Student	
Birthdate	Student	Is student of "drinking age"?
City	Home_Base	Hometown (given by city, state) is important for small talk, but full address not needed by dean
State	Home_Base	
Major	Student	Indicates what the student's doing in college besides hanging out
GPA	Student	How's student doing grade-wise
Factoid	Top_Scholar	Data to remember student by
Summer_Plans	Top_Scholar	Or other conversation topic

Figure 16.22 The Dean's View fields showing their source in physical database tables.

Join Three Tables into One

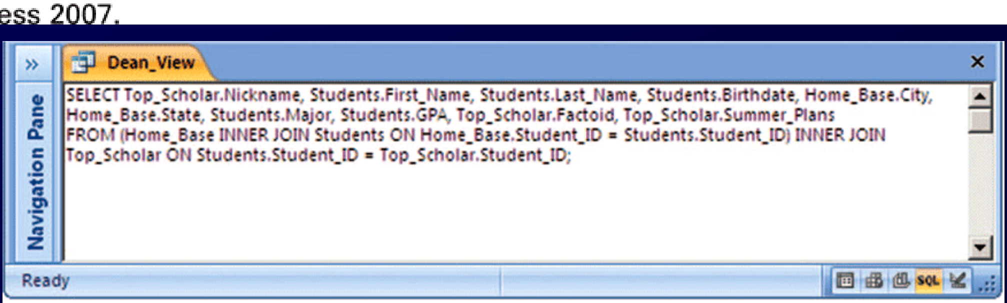
- Join using **Top_Scholar**, **Student**, and **Home_Base** tables matching on the Student_ID attribute across all three tables
- Trim the Table
  - Project – retrieve certain columns
- Join-then-trim strategy

Software Creates Dean's View



Field:	Last_Name	Birthdate	City	State	Major	GPA	Factoid	Summer_Plans	
Table:	Students	Students	Home_Base	Home_Base	Students	Students	Top_Scholar	Top_Scholar	
Sort:									
Show:	☑	☑	☑	☑	☑	☑	☑	☑	□
Criteria:									
or:									

Figure 16.23 The Query By Example definition of the Dean's View table as expressed in Microsoft Access 2007.



```
SELECT Top_Scholar.Nickname, Students.First_Name, Students.Last_Name, Students.Birthdate, Home_Base.City,
Home_Base.State, Students.Major, Students.GPA, Top_Scholar.Factoid, Top_Scholar.Summer_Plans
FROM (Home_Base INNER JOIN Students ON Home_Base.Student_ID = Students.Student_ID) INNER JOIN
Top_Scholar ON Students.Student_ID = Top_Scholar.Student_ID;
```

Figure 16.24 SQL query created for the Dean's View by the Query By Example data in Figure 16.22.

Structured Query Language

SQL

SELECT

- **SELECT** * **FROM** tablename;
  - Selects all fields from the table
- **SELECT** first_name, last_name, GPA
  FROM Students
  WHERE Student_ID = 0344567;
  - Selects first and last names, GPA for the student with ID of 0344567

KEYWORD	PURPOSE
SELECT	Identifies columns to be displayed
FROM	Identifies tables hold the needed data
WHERE	Limits the number of rows to be returned

SELECT Examples

-
  ```
  SELECT FName, LName
  FROM Student
  WHERE Major = "INFO";
  ```

SELECT Examples

- Select records with or without empty fields
  - **SELECT** LName
    FROM Student
    WHERE FName **IS NULL;**
  - **SELECT** LName
    FROM Student
    WHERE FName **IS NOT NULL;**

SELECT Examples

- Sorting query results
  - **SELECT** StudentID, LName
    FROM Student
    ORDER BY LName **DESC;**
    - Descending order Z-A, 9-0
  - **SELECT** StudentID, LName
    FROM Student
    ORDER BY LName **ASC**;
    - Ascending order A-Z, 0-9

JOIN Examples

- **SELECT** Student.FName, Student.LName, Advisor.LName
  FROM Student **INNER JOIN** Advisor **ON** Student.AdvisorID = Advisor.AdvisorID;
  - Joins the Student and Advisor tables and displays first and last names of all students and each student’s advisor by last name

Advanced Filtering

- Other ways to reduce the number of rows:

Operator	Symbol
Equals	=
Not equal	<>
Greater than	>
Less than	<
Greater than or equal to	>=
Less than or equal to	<=

Advanced Filtering Examples

-
  ```
  SELECT FName, LName
  FROM Advisor
  WHERE HireDate >= 1987;
  ```

-
  ```
  SELECT FName, LName
  FROM Student
  WHERE
      AdvisorID = 44232 AND
      Major = "INFO";
  ```

Aggregate Functions

- sum, avg, max, min, count, etc.
- W3 Schools: SQL Tutorial
  - Search for Aggregate Functions
  - http://www.w3schools.com/sql/sql_groupby.asp

Aggregate Functions

- **What is the total number of INFO majors?**

  -
    ```
    SELECT COUNT(Student_ID)
    FROM Student
    WHERE Major = "INFO";
    ```
- **What is the average GPA of INFO majors?**

  -
    ```
    SELECT AVG(Grade)
    FROM Student
    WHERE Major = “INFO”;
    ```