



Announcements

- Free copy of Access, Vista, etc., for educational/academic use:
 - Links on Computing page on Course Web site
 - Search for CSE or INFO to find the link on the page
 - Username is your full UW email address
 - Password is different!
 - Click on "send a reminder"
 - Check wherever your email forwards to
 - If you are in INFO100, send me an email.



A Table with a View (continued)

Primary keys, normalization, and SQL

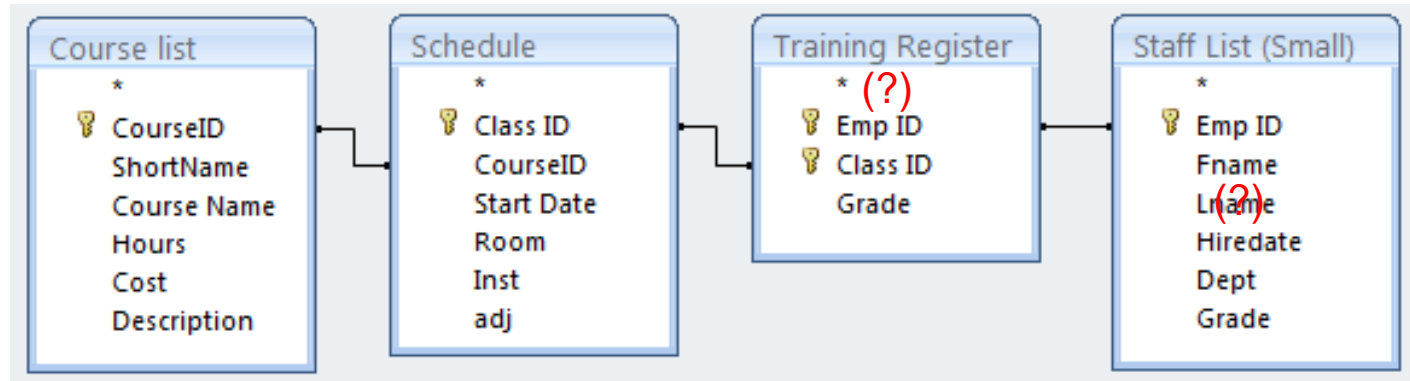


Video

- Primary Keys (5 min.)



Primary/Foreign Key



- 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)

Last Name	First Name	Calculated Field Full Name	Multipart Field City State Zip	Calculated Field Hourly	Calculated Field Weekly	Multivalued Field 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

Multivalued 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
(5 min.)



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		
<i>Name</i>	<i>Area</i>	<i>Elevation</i>
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	<i>Island Name</i>
area	Number	<i>Area in square kilometers</i>
elevation	Number	<i>Highest point on the island</i>

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 or 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 new tables
- The questions we ask of a database are answered with a whole new table, or view

Nations

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

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.



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:
 - Users see only what they need
 - Users see only what they have permission to see



Physical vs. Logical

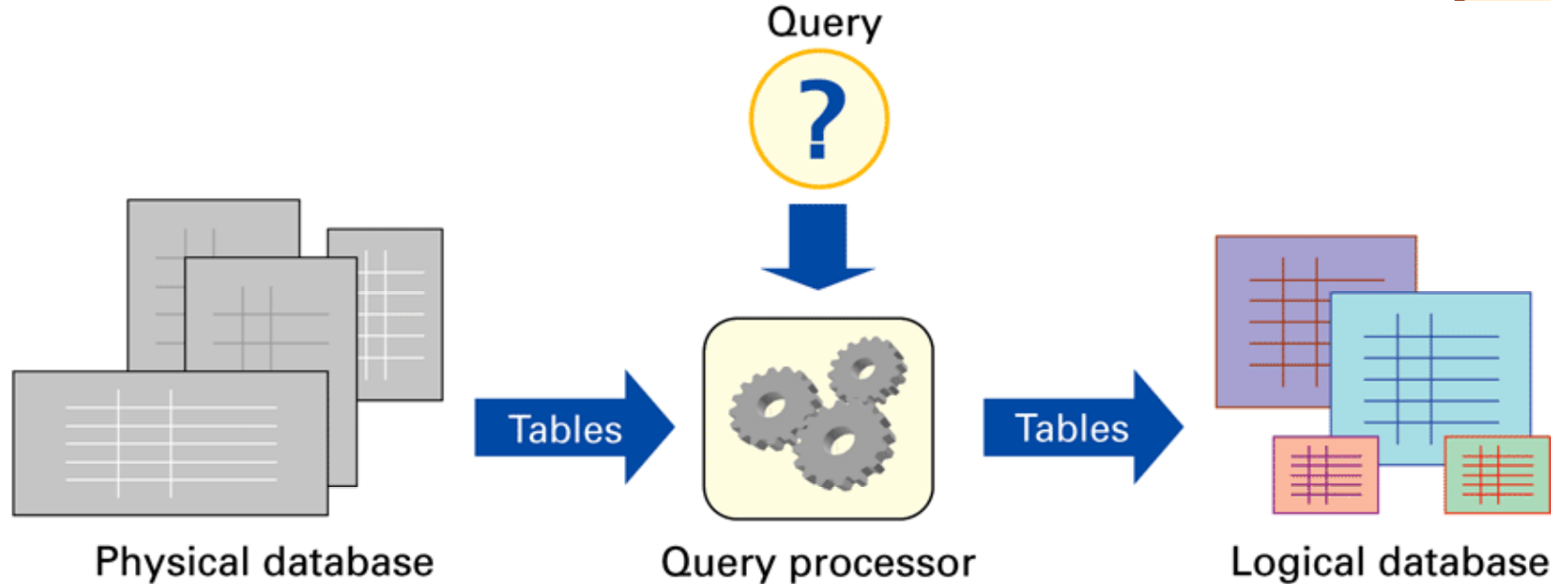


Figure 16.15 Structure of a database system. The physical database is the permanent repository of the data; the logical database, or view of the database, is the form of the database the users see. The transformation is implemented by the query processor, and is based on queries that define the logical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



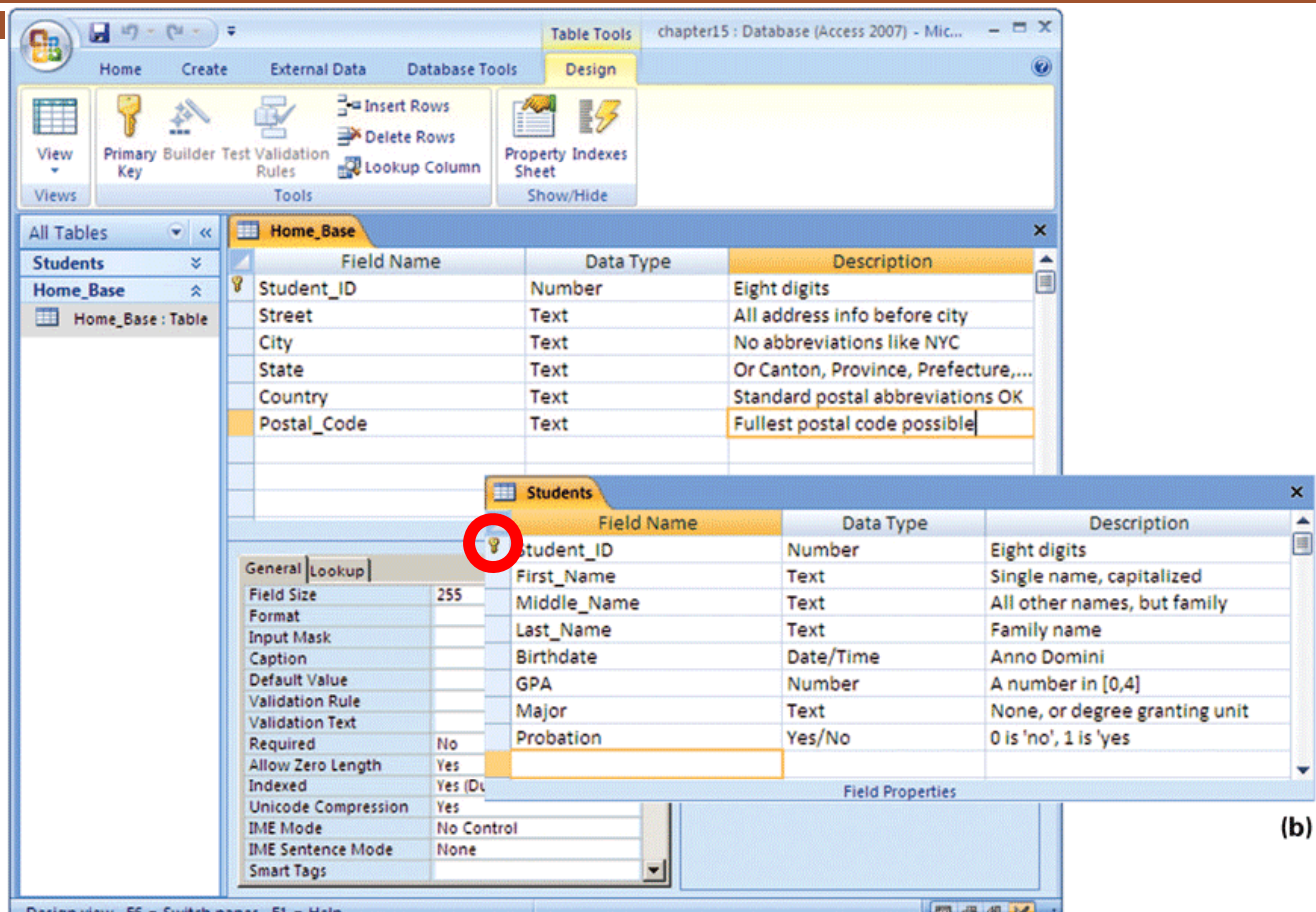
Logical Database

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



Defining Physical Tables

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



(a)

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.



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

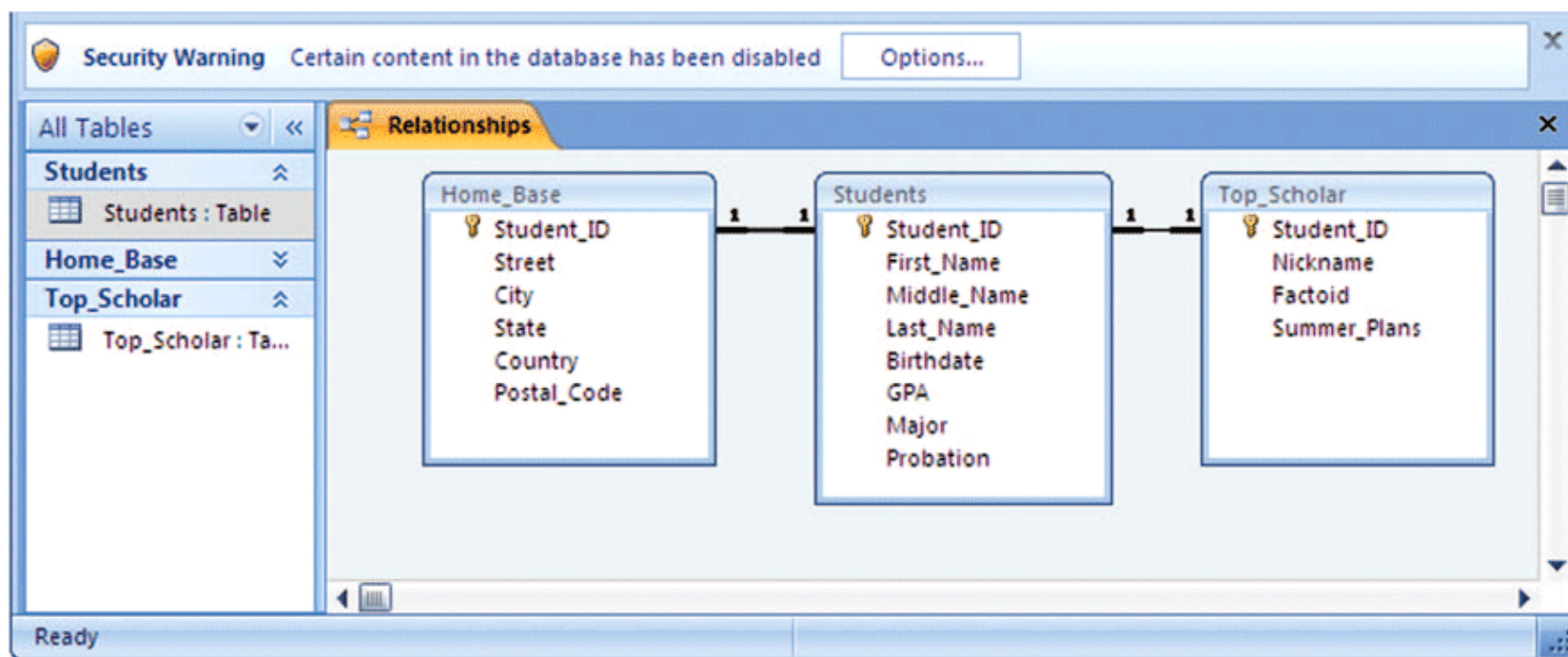


Figure 16.17 The *Relationships* window from the Microsoft Access database system; the 1-to-1 *Lives_At* and *Home_Of* relationships are shown between *Home_Base* and *Students*.



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.