# Relational Database Development

(152-156)

Database Design

| Notes | Activity |
| --- | --- |
| 1. Database Design Goals -- Database that is:
	* Adaptable
		+ Fields and tables can be added (removed) easily
	* Flexible
		+ Data can be retrieved in an unlimited number of ways
	* Accurate
		+ No data redundancy
		+ Validation on fields
		+ Default values
		+ Look ups
 |  |
| 1. Step 1 – Fact Finding
	* Determine field (data storage) requirements
	* Sources:
		+ Current users (owners)
		+ Existing databases
		+ Existing forms or other documents
	* Don’t worry about grouping, simply list
	* Split *multi-part* fields into separate fields
		+ Example: Split **Name** into **FirstName** and **LastName**
		+ Example: Split **Address** into **Street**, **City**, **State** and **Zip**
		+ Example: Split **Phone** into **AreaCode** and  **Phone**, maybe **Extension**
 | Handout Student Enrollment field list |
| 1. Step 2 – Name Tables
	* Browse through field list, list those tables that are obvious (others might (will) surface later)
 | List tables for Enrollment Database |
| * + Table Naming Conventions
		- Add the *tbl* prefix to each table name
		- Name tables using either plural nouns **or** singular nouns. Don’t mix with in a database.
			* E.g. tblCustomers, tblLocations, tblVehicles
			* E.g. tblCustomer, tblLocation, tblVehicle
			* Unique and descriptive
			* 2012: Lean towards plural nouns
		- Ensure abbreviations are clear to everyone, not just those involved in the project.
		- Brief, but complete
			* Use minimum words necessary
		- Don’t include database terminology: *Record, File, Table*
		- Don’t include adjectives that restrict data
			* Example: ~~Wisconsin Rapids~~ Employees, ~~Stevens Point~~ Employees
			* Results in duplicate structures. Structures (field lists) of both tables will be identical
 |  |
| * + Make a separate table for *multi-value* fields.
		- Example: a field named **Hobbies** might contain “bowling, fishing”
		- Create a separate Hobbies entity (each hobby will be listed as a separate record in this table)
		- Multi-value fields are difficult to search and nearly impossible to validate or sort.
		- Tip: if the field name is plural, it’s probably a multi-value field.
 |  |
| 1. Step 3 – Draw Entity Relationship Diagram

1MComputerLab* + Entity Relationship Diagram (ERD) is picture that shows the relationships between tables of a database
	+ Helps discover additional tables and defines relationships
	+ Rectangle used to represent each table in a database
	+ Line drawn between tables that are **directly** related
	+ At end of each line, include *cardinality*
		- One occurrence in table 1 is related to how many occurrences of table 2 (maximum number)
		- One occurrence in table 2 is related to how many occurrences of table 1 (maximum number)
		- For our purposes, the maximum is listed as 1 or many (M)
	+ The above ERD fragment expresses that:
		- “One lab contains (M)any computers”
		- “One computer exists in only one (1) lab”
 | Draw ERD for Enrollment |
| 1. Step 4 – Determine Primary Key for each Entity
	* The primary key is the field or fields whose value uniquely identifies a record in that table.
		+ For Lab above, it might be Room Number
		+ For Computer above, it might be ID Number
	* Primary keys can be a combination of two keys
		+ For Lab above, if the building has multiple floors, a combination key might be Room Number plus Floor (e.g. Room 10 on Floor 5)
	* If you need to combine 3 or more fields to create a unique primary key, consider creating an *ID Number* field for that table.
		+ These keys are usually *autonumber* fields
		+ Lately, I’ve been using these as the keys in almost all my tables.
	* Primary key requirements:
		+ Unique. No two keys will have the same value
		+ Cannot be *null*. In multi-field keys, none can be *null*
		+ Values in field rarely (if ever) change
 | Define keys for Enrollment database |
| 1. Step 5 – Resolve Many-to-Many Relationships
	* Many-to-Many (M-M) are relationships where the cardinality is M (many) in both directions.
		+ The Lab-Computer example above is a 1-M (one-to-many) relationship. The following represents a M-M relationship
		+ “One customer orders many products.”

ProductCustomerMM* + - “One product is purchased by many customers.”
	+ M-M relationships are nearly impossible to implement using a database program
	+ M-M relationships must be *resolved* into multiple 1-M relationships in order to implement the database
	+ Resolving M-M Relationships
		- Insert a new entity between the two entities
		- Name the new entity.
			* ”What is one occurrence of table1 combined with one occurrence or table2 called?”
			* ”One customer ordering one product is called…?an ordered product.”
		- Re-evaluate the cardinality of the new relationships
			* Probably 1----M [] M----1(Manys attached to new entity)

11MMOrderedProductCustomerProduct* + - Determine the primary keys (always at least 2) for the new entity.
			* Usually the keys from the two *parents**Parent entities* are those on the 1 side of a relationship (Customer and Product)*Child entities* are those on the M side of a relationship (Ordered Product)One entity can be the parent in one relationship and a child in a different relationship.
 | Resolve M-M relationships in Enrollment. |
| * + Another factor you should consider when defining table relationships (and should probably discuss with owners/users) is what happens to child records when parent records are deleted.
		- Restrict Delete
			* Parent record cannot be deleted until all child records (in all child tables) have been deleted.
			* Preferred technique. Requires consideration of affects of deleting this parent record.
		- Cascade Delete
			* When parent record is deleted, all associated child records (in all child tables) are automatically deleted.
			* Dangerous
 |  |
| 1. Step 6 – Determine the Linking Fields (*Foreign Keys*)
	* For every relationship, the primary key from the parent table must exist in the child table. This is what links the tables together in a relational database.
	* Often, the links will already exist because of M-M resolution.
	* If the parent’s primary key does not exist in the child, copy the field into the child table.
		+ This field DOES NOT become part of the child’s primary key.
		+ Designate the field as a link (L)
 | Copy keys from Program and Instructor into child tables. |
| 1. Step 7 – Remove Calculated Fields and Constants
	* Because today’s computers are so fast, it’s better to calculate these values as you need them instead of storing them in the database.
	* Additionally, if you calculate them as you need them, you ensure the values are always up to date.
	* Make a separate list of the calculated fields you removed. Include the equation used to calculate the value.
	* Ensure all the parts of the equations are stored somewhere in the database.
		+ Equation parts can be stored in different tables (linking allows you to bring them together)
	* If parts can be calculated, don’t store them either
	* Constants are fields that ALWAYS store the same value
		+ No need to waste storage space
		+ Print the constant value on reports when needed
	* There are exceptions to this rule. Values that rarely change, though calculated, may be fields in the database. I’ve never run into an instance of this though.
 | Remove GPA from Student table GPA = Total Points / Total CreditsTotal Points = Sum of all grade pointsTotal Credits = Sum of all credits earned✓ Grade Points available (determined from letter grade)* Credits Earned available

Remove State (constant)Remove City, create ZipCity table to lookup city based on zipZip is linking field in Student |
| 1. Step 8 – Assign Remaining Fields to Entities
	* For all remaining fields (from Step 1), assign to one **and only one** table.
	* Only linking fields may be duplicated in a database.
 | Assign fields to entities in Enrollment database. |
| * + Field Naming Standards
		- Apply to primary keys and linking fields as well.
		- Use singular nouns
			* If plural makes more sense, this is not a field but another table (see [multi-value fields](#multivalue) above).
		- Unique and descriptive
			* Include table name when field name occurs in two tables (StudentAddress, InstructorAddress)(optional)
		- Use minimum number of words
		- Use acronyms and abbreviations wisely (only if everyone understands them)
		- If the name includes “/” “&” “-“ “and” “or”, it probably represents two or more fields. Split them.
		- Split multipart fields into separate fields
			* If a field can be decomposed into *parts*, it’s probably more than one field.
			* Example: *Address (street, city, state, zip) Phone (area code, number, extension)*
 |  |
| 1. Step 9 – For all Fields, Determine Type (And Size)
	* Use types and sizes available in your database program
	* Types and sizes of **linking fields** must be identical in each table
	* For Access:
		+ Short Text
			- Most common type of field
			- Determine appropriate maximum size for field (255 characters max)
			- If you’re not sure, estimate on the low side
				* Easier to expand a field than to remember to shrink
			- Example: T20 (short text, 20 characters)
		+ Long Text (Memo)
			- Text–type field that can store up to 65,000 characters
			- Example: M
		+ Number
			- Access has five different types of Numbers
			- Select the appropriate type to use memory/storage effectively
				* Byte (whole number, 0-255)
				* Integer (whole number, ±32,767)
				* Long Integer (whole number, ±2.1 billion)
				* Single (number with decimals, 7 digits of accuracy after decimal point)
				* Double (number with decimals, 15 digits of accuracy after decimal point)RARELY USED!
			- Example: B I L S or D
		+ Date/Time
			- Stores a combination of date and time or any part thereof
			- Example: D/T
 | Assign types and sizes to all fields. |
| * + - Currency
			* Stores a dollar amount
			* Usually better to use Single from above
			* Example: $
 |  |
| * + - Yes/No
			* Stores a boolean value: yes/no, true/false
			* Example: Y/N
		- Hyperlink
			* Stores a web address or email address
			* Example: H
		- OLE Object
			* Stores, or links to, an image, sound clip or video clip
 |  |
| * + Designate the *logical keys*
		- In tables that use autonumber fields, you should (with the user/owner’s help) determine which fields in the table should not allow duplicate values.
		- The autonumber field will ensure the primary key will not have duplicate values
		- If the primary key is NOT an autonumber field, it should always be designated as a logical key. Your program must ensure the field is not a duplicate to prevent the database from causing an error.
		- But, there are usually other fields in the table that should not contain duplicates.
		- In a movie database, you might include a autonumber primary key, but the database should not allow duplicate entries for **movie title** combined with **release date**
		- Some tables may allow duplicate records but only with user permission.
			* In a student table (with autonumber primary key), your GUI program might allow two (or more) students with the same first, middle, last name combination—**but only with user permission**.
			* These types of tables are fairly rare. Most tables have logical keys that are never duplicated.
 |  |
| * + Logical Keys: Rules and Guidelines
		- Every table has logical keys
		- Autonumber fields are NEVER logical keys
		- Primary keys that not autonumber are ALWAYS logical keys
		- Linking table linking fields are always logical keys
		- Think about all other fields, should duplicates be allowed? If no—logical key
		- Look for combinations of fields. Should their concatenated values allow duplicates? If no-logical keys.
 |  |
| 1. Step 10 – Ensure No Redundancy Except for Linking Fields
	* Check for *synonyms*, two fields with different names that are actually the same thing.
		+ Example: Social Security Number and Employee ID
	* Double-check to ensure non-linking fields only occur in one entity
 |  |
| * + Field Formatting / Validation Considerations
		- # designate digits required for text field
		- Use a lookup for this field

* + - * All linking fields should be lookups
		- AutoCap: automatically capitalize the first letter of each word in the field
		- Uppercase: Automatically capitalize all letters in the field
		- n1 – n2 : numeric value range check
		- Auto Populate from *field*Automatically populate this field from another field in the database (credits earned = current credits).
			* Not a lookup
			* User not usually allowed to edit
		- Required. This is a required field (not a big fan of this)
			* Keys are automatically required
 |  |
| 1. Final Thoughts
	* Database design is best done by a group of people unless you have significant experience.
	* Don’t be afraid of undiscovered errors in your design
		+ When you build the database, errors will surface and you can correct them early
		+ When you populate the tables with data, other errors might surface. Again, you’ll usually catch these early on.
		+ If you follow these guidelines, your database will be *adaptable, flexible* and *accurate.* Any design errors you find after using the database for a while (lots of data entered) should still be relatively easy to correct, especially with Access’ help.
 |  |