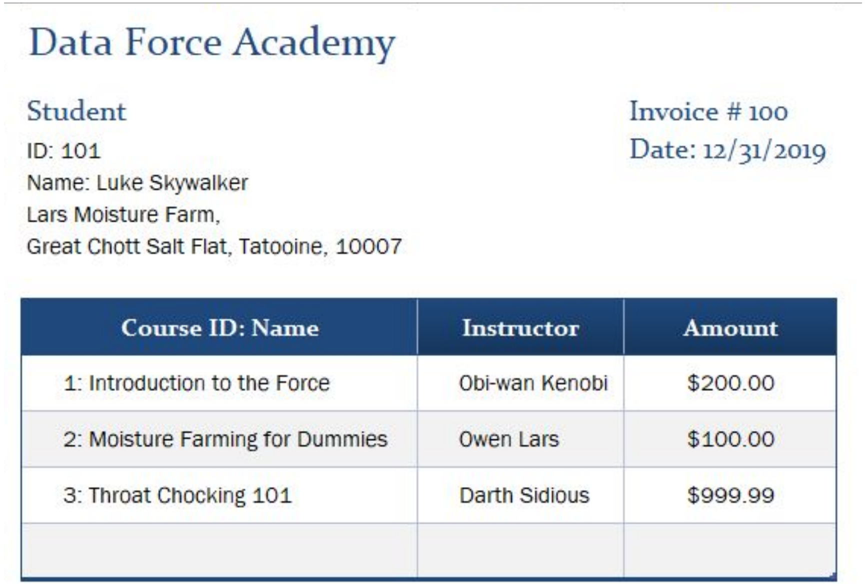
Normalization

Relational Database Modelling

- The good, the bad, and the ugly
  - Bad designs can happen, do happen, and will always happen, a lot.
  - Bad designs make it difficult to use the database, and increase the cost of usage.
  - Improving bad design is a highly skilled and well paid job -- It is called Data Architect.

Original Student Registration Form

- Has all information in the same form



Data Force Academy

Student

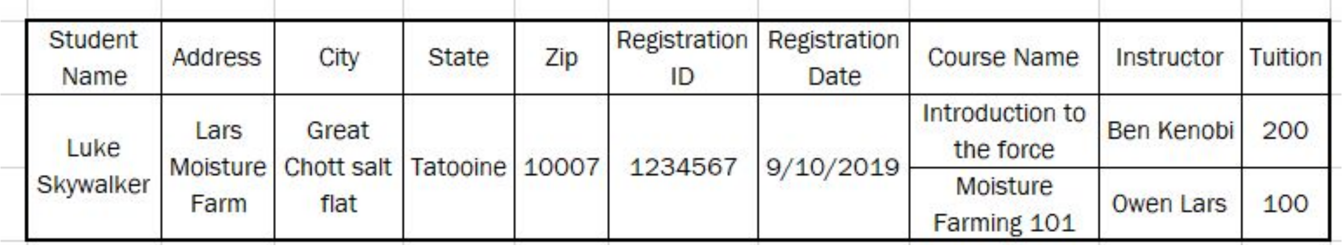
ID: 101
Name: Luke Skywalker
Lars Moisture Farm,
Great Chott Salt Flat, Tatooine, 10007

Invoice # 100
Date: 12/31/2019

Course ID: Name	Instructor	Amount
1: Introduction to the Force	Obi-wan Kenobi	$200.00
2: Moisture Farming for Dummies	Owen Lars	$100.00
3: Throat Chocking 101	Darth Sidious	$999.99

Student Registration Table Design

- Designed directly from the original registration form



Student Name	Address	City	State	Zip	Registration ID	Registration Date	Course Name	Instructor	Tuition
Luke Skywalker	Lars Moisture Farm	Great Chott salt flat	Tatooine	10007	1234567	9/10/2019	Introduction to the force	Ben Kenobi	200
							Moisture Farming 101	Owen Lars	100

- Shorthand Representation

```
Registration (Student ID, Student Name, Address, City, State, Zip,
Registration ID, Registration Date, (Course ID, Course Name,
Instructor Name, Tuition))
```

Issues with This Design

- Two key issues
    - Non-atomic value: repeating group, nested record
    - Duplication of information
- Both are functional dependency issues

Non Atomic Value

- Repeating group: Multiple course inside same registration tuple

  Registration (Student ID, (CourseID, Course Name))

- Nested value: Sub tuple inside tuple

  Registration (Student ID, (CourseID + Course Name))

  ```
  “DS530 - Big Data and Data Management”
  ```

Issues with Non Atomic Value

- Some values should be determined by the combination of keys
- Must go inside the sub-relation in the relation for processing
- Not manageable by Relational model and SQL

Issues with Duplicated Data

- Unnecessary dependency on non key columns
- Same operation must be performed in multiple locations.
  - Performance hit.
  - Inconsistency may happen if missing an update.

Normalization

- Normalization: The process of structuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity. (“Database Normalization”, *Wikipedia*)

Normalization Forms

- Normalization is defined with Normal Form (NF), a formal set of rules to check against a table
    - Table is xxNF compliant if it meets certain criteria and achieves certain level of dependency properties

Normalization Forms

- Multiple levels of normal form exist, from 1NF to 6NF, with two more derived forms.
  - A higher level of database normalization cannot be achieved unless the previous levels have been satisfied.
    - Must be 1NF to be 2NF. Must be 2NF to be 3NF. etc.
  - Usually 3NF is good enough for real world applications.
    - That’s what we are going to teach in this class.

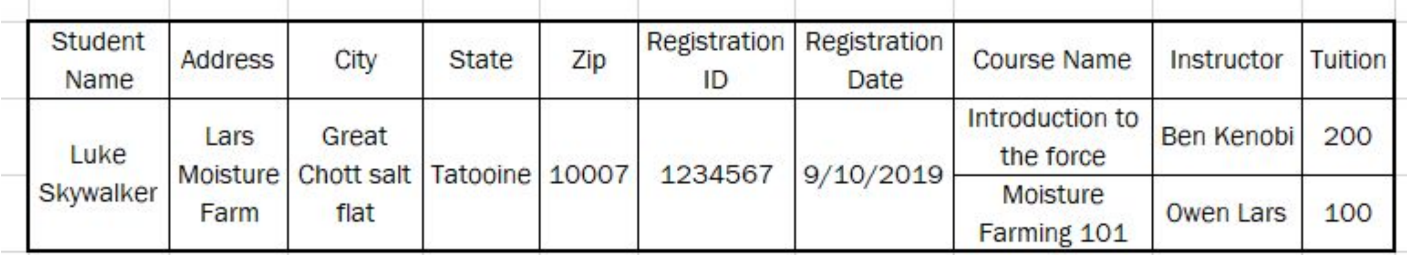
Normalization Process

- Normalization is generally achieved by “decomposition”: The process of breaking up or dividing a single relation into two or more
    - As the relation has fewer columns, its functional dependency becomes simpler to handle.
    - Starting from 1NF, decompose to one normal form at a time:
        - 1NF -> 2NF -> 3NF

Normal Forms

Non Atomic Value

- Non atomic value in a table definition



Student Name	Address	City	State	Zip	Registration ID	Registration Date	Course Name	Instructor	Tuition
Luke Skywalker	Lars Moisture Farm	Great Chott salt flat	Tatooine	10007	1234567	9/10/2019	Introduction to the force	Ben Kenobi	200
							Moisture Farming 101	Owen Lars	100

- Shorthand Representation

Registration (Student Name, Address, City, State, Zip, Registration ID, Registration Date, (Course Name, Instructor Name, Tuition))

First Normal Form

- If there is no repeating group or nested value, the table is in 1NF.
- A table is in First Normal Form (1NF) when
    - each column of a table must have a single value. ("Database Normalization", *Wikipedia*)

First Normal Form

- Decomposition: Break up the non atomic value by pushing dependency on the primary key of original table into the non atomic value,
    - break the repeating group into separate rows, with the primary key of original table. The new primary key will be the original primary key plus the repeating group’s primary key.
    - break the nested value into separate columns, The new primary key will be the original primary key plus the nested entity’s primary key.

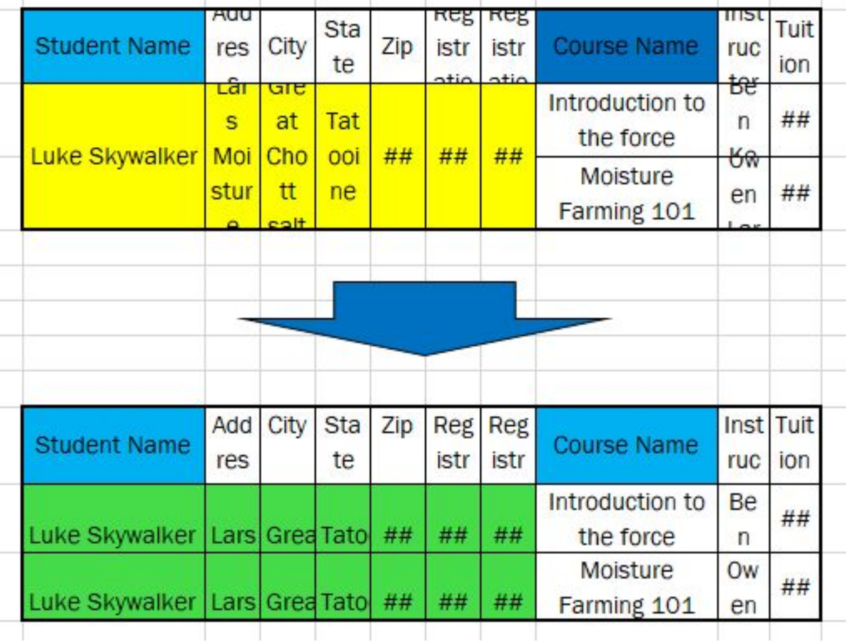
Decomposition to 1NF

Registration (Student Name, ...,

(Course Name, Instructor, Tuition))

Registration (Student Name, ...,

Course Name, Instructor, Tuition)



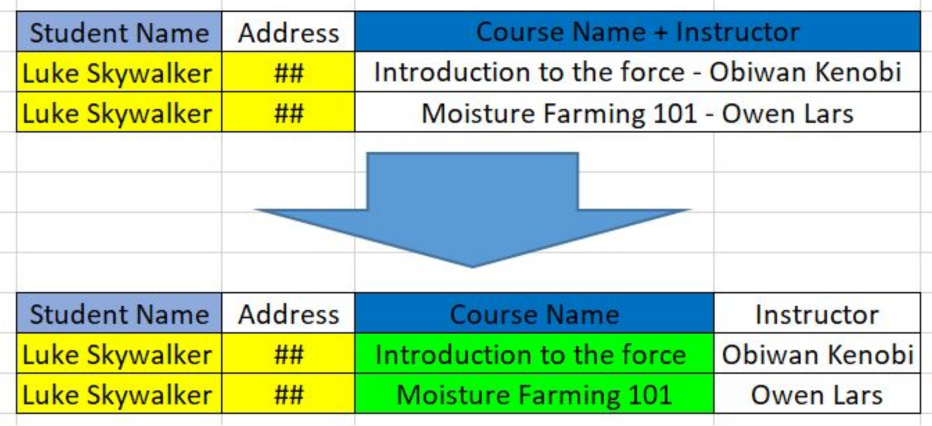
Student Name	Addres	City	State	Zip	Registratio	Registratio	Course Name	Instructor	Tuition
Luke Skywalker	Lars Moisture	Great Chott salt	Tatooine	##	##	##	Introduction to the force	Ben Ke	##
							Moisture Farming 101	Owen Lar	##

Student Name	Addres	City	State	Zip	Registr	Registr	Course Name	Instruc	Tuition
Luke Skywalker	Lars	Grea	Tato	##	##	##	Introduction to the force	Ben	##
Luke Skywalker	Lars	Grea	Tato	##	##	##	Moisture Farming 101	Owen	##

Decomposition to 1NF

Registration (Student Name, ..., (Course Name + Instructor))

Registration (Student Name, ..., Course Name, Instructor)

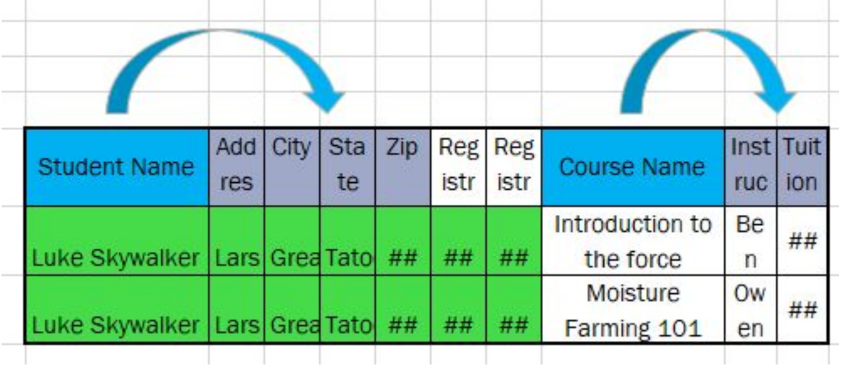


Student Name	Address	Course Name + Instructor
Luke Skywalker	##	Introduction to the force - Obiwan Kenobi
Luke Skywalker	##	Moisture Farming 101 - Owen Lars

Student Name	Address	Course Name	Instructor
Luke Skywalker	##	Introduction to the force	Obiwan Kenobi
Luke Skywalker	##	Moisture Farming 101	Owen Lars

Partial Dependency

- What if the primary key is a combination of columns, and some columns are only dependent on one of the columns?
    - Partial dependency: Address/City/State/Zip are functional dependent on Student Name. So they repeats for each row of that student.



Student Name	Addres	City	State	Zip	Registr	Registr	Course Name	Instruc	Tuition
Luke Skywalker	Lars	Grea	Tato	##	##	##	Introduction to the force	Ben	##
Luke Skywalker	Lars	Grea	Tato	##	##	##	Moisture Farming 101	Owen	##

Second Normal Form

- A table without partial dependency is in 2NF
- A table is in Second Normal Form (2NF) when
  - it is in 1NF, and
  - every non-key attribute must depend on the whole key, not just part of it. (“Database Normalization”, *Wikipedia*)
- Single column primary key + 1NF = 2NF

Second Normal Form

- Decomposition: Break up each group of dependencies into a new relation.
  - Each partial dependency is a new relation
  - The original dependency on the compound key is another relation
    - Don’t miss out any dependency, especially when there are multiple dependencies: Partial key itself has a dependency, then it is also a part of the compound key dependency.

Decomposition to 2NF

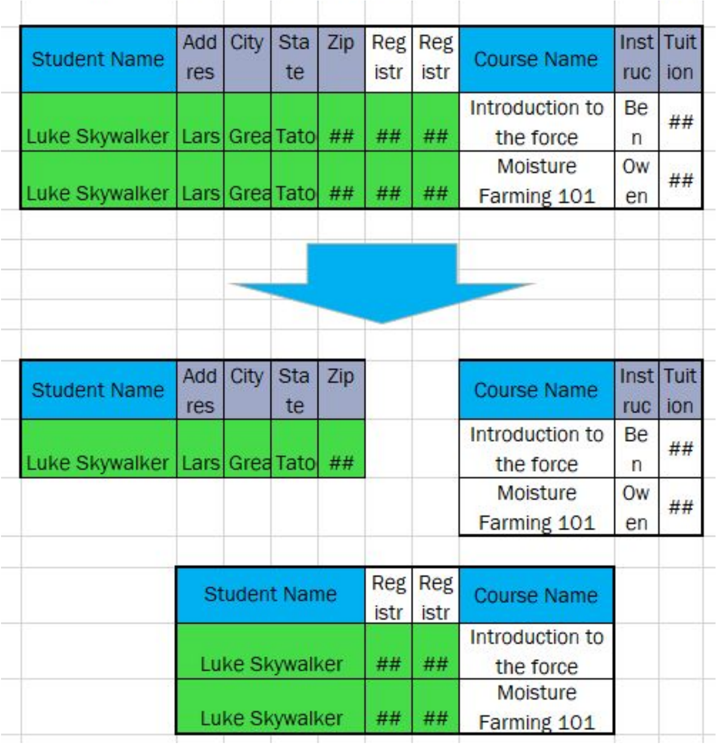
- Break up partial dependency

Registration (Student Name, Address, City, State, Zip, Registration ID, Registration Date, Course Name, Instructor, Tuition)

Student (Student Name, Address, City, State, Zip)

Course (Course Name, Instructor, Tuition)

Registration (Student Name, Registration ID, Registration Date, Course Name)



Student Name	Add res	City	Sta te	Zip	Reg istr	Reg istr	Course Name	Inst ruc	Tuit ion
Luke Skywalker	Lars	Grea	Tato	##	##	##	Introduction to the force	Be n	##
Luke Skywalker	Lars	Grea	Tato	##	##	##	Moisture Farming 101	Ow en	##

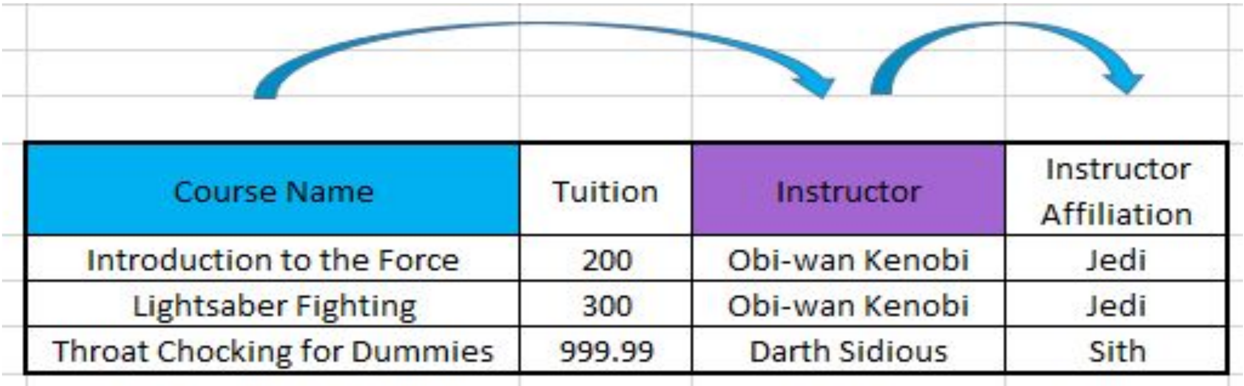
Student Name	Add res	City	Sta te	Zip
Luke Skywalker	Lars	Grea	Tato	##

Course Name	Inst ruc	Tuit ion
Introduction to the force	Be n	##
Moisture Farming 101	Ow en	##

Student Name	Reg istr	Reg istr	Course Name
Luke Skywalker	##	##	Introduction to the force
Luke Skywalker	##	##	Moisture Farming 101

Transitive Functional Dependency

- What if column A is functional dependent on a non-key column B, and B is in turn functional dependent on another column C?
  - Transitive functional dependency. Redundancy in column C.



Course Name	Tuition	Instructor	Instructor Affiliation
Introduction to the Force	200	Obi-wan Kenobi	Jedi
Lightsaber Fighting	300	Obi-wan Kenobi	Jedi
Throat Chocking for Dummies	999.99	Darth Sidious	Sith

Third Normal Form

- A table without transitive dependency is in 3NF.
- A table is in Third Normal Form (3NF) when it
    - is in 2NF, and
    - has no transitive dependencies.

Third Normal Form

- Decomposition by breaking up each dependency into a new relation
    - Transitive key and its dependencies go into another relation.
    - The original primary key and its dependencies (including transitive key column) must stay in a relation, i.e., transitive key column must remain in original table.

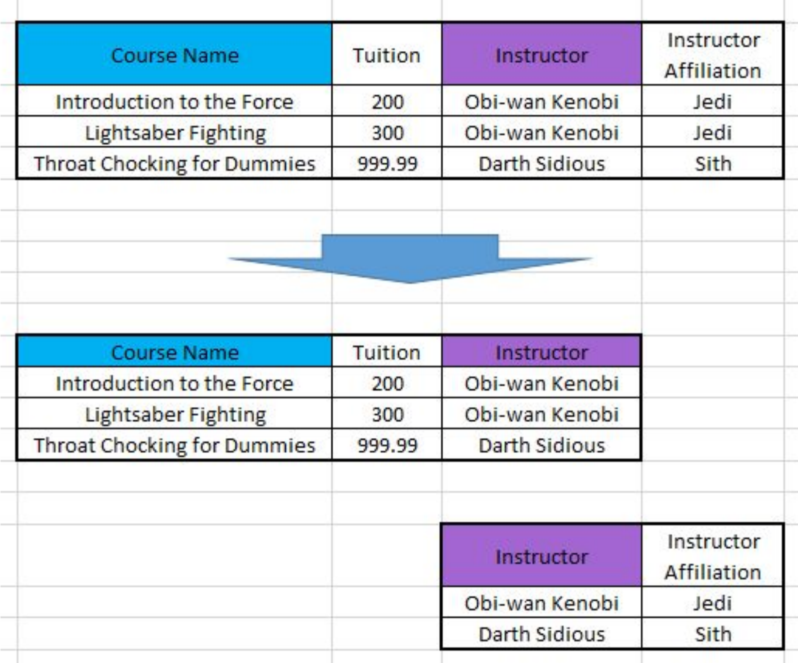
Decomposition to 3NF

- Break up transitive dependency

Course (Course Name, Tuition, Instructor, Instructor Affiliation)

Course (Course Name, Tuition, Instructor)

Instructor (Instructor, Instructor Affiliation)



Course Name	Tuition	Instructor	Instructor Affiliation
Introduction to the Force	200	Obi-wan Kenobi	Jedi
Lightsaber Fighting	300	Obi-wan Kenobi	Jedi
Throat Chocking for Dummies	999.99	Darth Sidious	Sith

Course Name	Tuition	Instructor
Introduction to the Force	200	Obi-wan Kenobi
Lightsaber Fighting	300	Obi-wan Kenobi
Throat Chocking for Dummies	999.99	Darth Sidious

Instructor	Instructor Affiliation
Obi-wan Kenobi	Jedi
Darth Sidious	Sith

Boyce-Codd Normal Form

- Boyce-Codd Normal Form (BCNF): An enhancement of 3NF
    - All non-trivial dependencies are on table key
        - “Trivial” here means self-referencing. It only happens when there are multiple candidate keys, which is very rare.
- In real world, most “3NF compliant” designs are already “BCNF compliant”.
- We will use the term 3NF to reference both NFs in this class.

Review of Normalization

- Follow different NFs: 1NF -> 2NF -> 3NF
  - 1NF: Each element has a key
  - 2NF: Dependency is on a whole key
  - 3NF: All dependencies are only on the key

Denormalization

- Normalization may cause performance issue
  - Questions on Registration: Must read from multiple tables (JOIN)
- Denormalization to put groups of attributes as one block
  - State in an address record: Zip -> State
  - Unstructured/Semi-structured data
  - Data warehouse (Query and Reporting)
  - NoSQL DB

Physical Design and Relational Database

Physical Design

- Physical Design is a relational model process and not related to Entity Relationship Model
  - Both Entity and Relationship are relations so they follow the same design process.
  - Same design principles apply to other non-ER Model relational designs.
  - Still, ER Model is the most popular relational model out there. We will not talk about other models in this class.

Mapping Relation in Relational DB

- Relational DB is designed to best represent relational models
  - Relations are directly mapped to 2-dimensional tables
  - Distinct instance of the relation = Row in table
    - Each row represents a distinct instance of the relation.
  - Attribute of relation = Column of rows
    - Each column represents an attribute of the relation.

Tables in Relational DB

- All tables are 2-dimensional
    - Each table is a set of rows with the same columns
    - Each row must be unique: Dups provide no new information.
    - Each column has its own name and data type, and is the same across all rows.

Database Representation

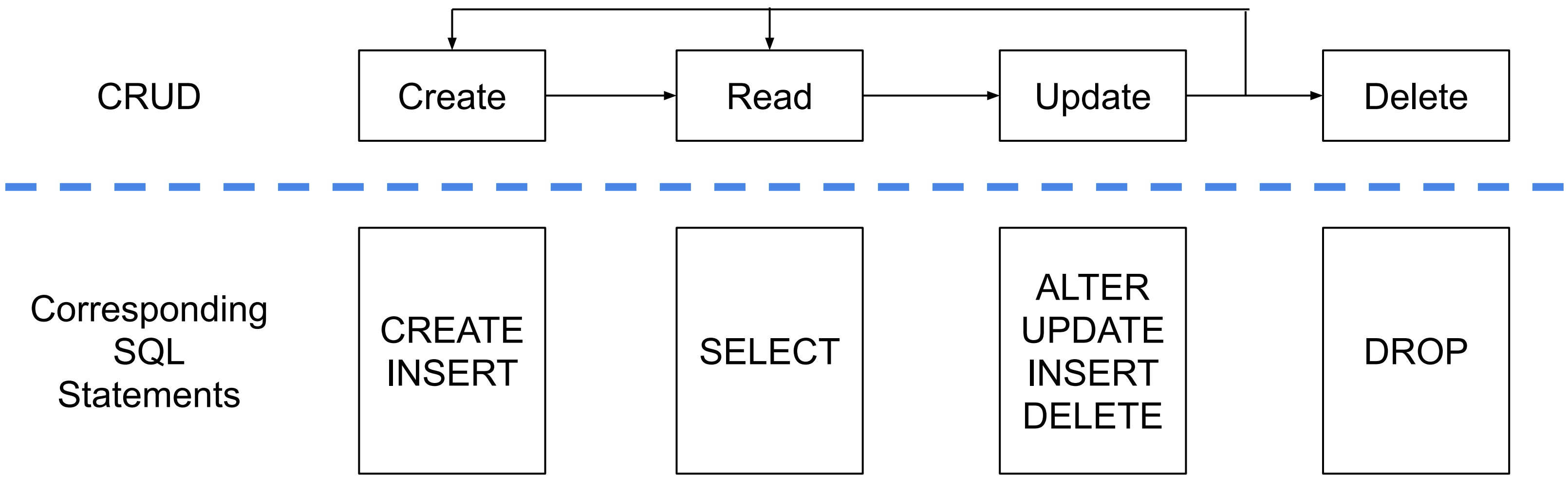
- Entity/Attribute: One entity per table, and attributes are columns
  - Student: Student ID, Student Name
- Relationship: One relationship per table. Entity keys and relationship attributes are columns.
  - 1:M relationships can usually be merged into functional determinant entity table

Changing Table Data

Data Lifecycle

How data exists in computer systems: Create, Read, Update, Delete (CRUD)

- Generic lifecycle of data in ANY computer system, not SQL specific.



CRUD	Create	Read	Update	Delete

Corresponding SQL Statements	CREATE
	INSERT
		SELECT	ALTER
			UPDATE
			INSERT
			DELETE
				DROP

Correct Data Mistakes

Three approaches

1. DROP, re-CREATE, then re-INSERT
2. DELETE, then re-INSERT
3. UPDATE

DELETE Statement

- Delete rows meeting the WHERE condition.
  - Keyword: `DELETE FROM`, followed by the table name
  - Followed by the `WHERE` clause
    - Delete all rows if WHERE clause not specified --- Can have serious consequence!

```
DELETE FROM <table_name>
WHERE <boolean_expression>
```

Example DELETE Statement

```
DELETE FROM Instructor
WHERE Instructor_Affiliation = ‘Jedi’
```

- Going through all rows in the table. Delete the row if it meets the filtering condition.

INSERT Statement Review

- Single INSERT

```
INSERT INTO <table_name> (<column_list>)
VALUES (<value_list>)

INSERT INTO <table_name> VALUES (<value_list>)
```

Bulk INSERT from Other Tables

- Bulk Insert from another table:
  - SELECT clause must have matching columns in the target table
  - The INSERT column list (highlighted) is optional, which defaults to the column list in CREATE statement.

```
INSERT INTO <target_table_name> (<column_list>)
SELECT <column_list>
FROM <source_table_name>
WHERE <boolean_expression>
```

Simple UPDATE Statement

- Update all rows meeting the WHERE condition
  - Keyword: `UPDATE`, followed by table name
  - Keyword `SET`, followed by a list of column - value assignment
  - Followed by `WHERE` clause

```
UPDATE <table_name>
SET
    <column_name_1> = <value_1>
    , <column_name_2> = <value_2> ...
WHERE <boolean_expression>
```

Example UPDATE Statement

```
UPDATE Instructor
SET
    Instructor_Name = ‘Darth Vader’,
    Instructor_Affiliation = ‘Sith’
WHERE Instructor_Name = ‘Anakin Skywalker’
```

- Going through all rows in the table. Update the row if it meets the filtering condition.

UPDATE with Computation

- Value can be calculated from other fields

```
UPDATE Registration SET
   Actual_Tuition =
      Actual_Tuition * 1.1 -- 10% increase
WHERE Course_Name = ‘Light Saber 101’
```

Changing Table Structure

Correct Table Structure Mistakes

Two approaches

1. DROP, re-CREATE, then re-INSERT
2. ALTER TABLE

ALTER TABLE

- ALTER TABLE ADD/DROP
  - Keyword: `ALTER TABLE`, followed by table name
  - Keyword: `ADD` or `DROP`, followed by new column definition

```
ALTER TABLE Registration
ADD Discount DECIMAL(7,2)
```

ALTER TABLE … ALTER

- ALTER TABLE ALTER <column> TYPE <data type>

```
ALTER TABLE Registration
ALTER Actual_Tuition
TYPE DECIMAL(8,2)
-- Was DECIMAL(7,2)
```

Note: Some DBMS uses “ALTER TABLE MODIFY <column> <data type>”

ALTER TABLE … ALTER

- ALTER TABLE Column constraints: Need to specify a constraint name

```
ALTER TABLE tbl ADD CONSTRAINT constraint_name UNIQUE
(column_name);
ALTER TABLE tbl DROP CONSTRAINT constraint_name;
```

Comparing DROP, DELETE, ALTER

- DROP removes the table definition
  - No column list or WHERE
- DELETE removes rows.
  - After DELETE without WHERE, table still exists, just empty
  - Performed on rows, so no column list but can have WHERE
- ALTER TABLE … DROP removes a column.
  - Works on column so no WHERE

Comparing ALTER and SELECT

- ALTER ADD: Actually changes the table structure.
- SELECT AS: Returns calculated field but table remains the same.

Existence Checking

- PostgreSQL allows you to check the existence of tables before CREATE and DROP

```
CREATE TABLE IF NOT EXISTS T1 (ID INTEGER);
DROP TABLE IF EXISTS T1;
```

- Be careful of the potential side effect: You may change the table without knowing

NULL

NULL

- What is NULL: a special value to mark missing values, either non-existing or unknown
  - Example: mid name, home phone
  - NULL is represented using keyword NULL (case insensitive)
  - It is not: 0, empty string (‘’), ‘NULL’

NOT NULL

- Can a primary key like Student ID be NULL?
- Can “Emergency Contact Number” be NULL?
- NOT NULL in column definition
  - After data type, before comma
  - Can have multiple NOT NULL in one table
  - Default is NULLable if without NOT NULL
  - Primary key is automatically NOT NULL

NULL/NOT NULL Constraint

- Append `NULL` or `NOT NULL` to column definition

  ```
  Instructor_AFFLIATION CHAR(100) NOT NULL,
  ```

- Default is `NULL` if not specified.
- ALTER TABLE NULL/NOT NULL

```
ALTER TABLE tbl ALTER COLUMN col_name SET NOT NULL;
ALTER TABLE tbl ALTER COLUMN col_name DROP NOT NULL;
```

INSERTing NULL

- INSERT NULL by missing out column names

```
INSERT INTO Instructor (Instructor _Name)
VALUES ('Owen Lars');
```

- INSERT directly using the reserved word NULL (Don’t use ‘NULL’)

```
INSERT INTO Instructor
VALUES ('Owen Lars',NULL);
```

NULL Resulting from INSERT

- If the column has a DEFAULT value, DBMS will insert the default value instead of NULL.

```
INSERT INTO Instructor (Instructor _Name)
VALUES ('Owen Lars');
```

- NOT NULL columns (including PRIMARY KEY) MUST be in the column list of INSERT statement. Otherwise the INSERT statement will result in an error.

SET NULL in UPDATE

- Can set a value to NULL in UPDATE statement

```
UPDATE Registration SET
   Actual_Tuition = NULL
WHERE Course_Name = ‘Throat Choking 101’
-- This course is no longer offered in our school
```

Calculation Involving NULL

- Any calculation on a NULL value results in NULL

WHERE Clause: NULL

- Boolean logic with NULL: TRUE, FALSE, UNKNOWN
- Any logical operation with NULL returns UNKNOWN

```
1 > NULL
```

- NULL is neither equal to nor not equal to NULL

```
NULL = NULL

NULL <> NULL
```

NULL Detection

- NULL detection

```
Last_Name = NULL ✖  -- #1 common mistake
```

  - ‘= NULL’ and ‘< > NULL’ will always return UNKNOWN.
    - The worst issue is that it won’t show any error.
  - Use IS NULL and IS NOT NULL to detect NULL.