REMEMBER SEMICOLONS ON STUFF

# 

Objective-type: multiple choice, fill in the blank, match the following

ER - mostly objective-type, and a written response

SQL DML - SELECT - mostly objective-type, 1 written response. Not using delete insert and update.

No normalization

# Week 3

A% - starts with an A

%A - ends with an A

%A% - contains an A

\_ - (ex.) h\_t would return hot, hat, hit, etc.

HAVING - filters results from an aggregate function (count, max min, sum)

(ex.) select SUM from thing group by n having sum > 1000;

# Week 4

There is no “FOR ALL” in SQL so use two negations instead, “does not exist a part that the supplier does not supply”

Delete - delete from <relation> where <condition>

* Delete from <relation> deletes all tuples

Update

* Update <relation> set <list of assignments> where <condition>;
* Update sp set qty = 150 where qty = 100;

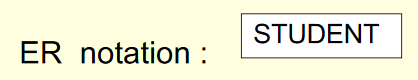
# Week 5

ER Model - Entity Relationship Model

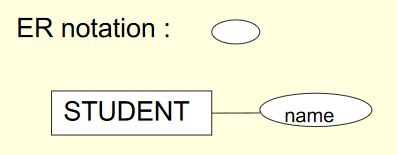
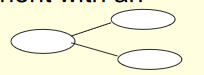
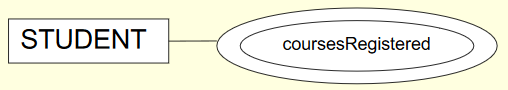
Main phases of design:

1. Requirements gathering
2. Conceptual Model
3. Logical Model
4. Physical Model

Entity

* An object in the real world with independent existence
* Is distinguishable from other objects
* (ex.)
  + A student in UofG can take several courses but is taught by just one instructor at any given point in time
  + Entities:
    - Student
    - Course
    - Instructor
* Denoted with a box around the entity
* 

Attribute

* Properties possessed by all members of an entity set
* 
* Simple attribute: Single component with an independent existence denoted as above
* Composite attribute: Attribute composed of multiple components, each with an independent existence
  + (ex.) address can be split into house number, street, city, postal code
  + 
* Classification:
  + Single value is the single circle denoted by simple attribute
  + Multi-valued holds multiple values for each occurrence of an entity
    - (ex.) courses of a student, as a student can take more than one course
    - 
  + Key attribute
    - Attribute whose values are distinct for each individual entity in the collection
    - Denoted by the circle with the attribute underlined

Study relationships in ER

# Week 6

All goddamn relationship stuff

Unary relationship - One entity with the relationship (ex.) A person is married to another person

Binary Relationship - Relationship between two entities (ex.) publisher publishes a book

Ternary Relationship - Three entities in the relationship (ex.) Teacher teaches a subject and teaches a student

Cardinality Ratio - Max number of relationships an entity can participate in

* (1:1) One to One
* (1:N) One to Many
* (N:1) Many to One
* (M:N) Many to Many

# Normalization

A technique for producing a set of relations with desirable properties (minimum data redundancy), given the data requirements of an enterprise. First developed by E.F. Codd (1972). Normalization is the process of converting relations into relations in good normal forms.

### Goals

* Minimize data redundancy
* Each property is desirable
* Remove update anomalies

Normalization involves the idea that we should be able to convert bad relations into good ones. Relations are classified into classes called [normal forms](#_z7losmtljrj4)

As normalization proceeds, relations become progressively more restricted (stronger) in format and also less vulnerable to update anomalies.

Normalization removes update anomalies by decomposing the relations into smaller relations based on the constraints.

## Drawbacks of Bad Design

Poorly designed relations contain redundant data, which leads to **update anomalies**

### Update Anomalies

Can you add a row where part of the key is NULL? **No.**

Should you delete a row where there is other relevant information? **No.**

### Insertion Anomalies

An independent piece of information cannot be recorded in a relation unless an irrelevant information must be inserted together at the same time.

### Modification Anomalies

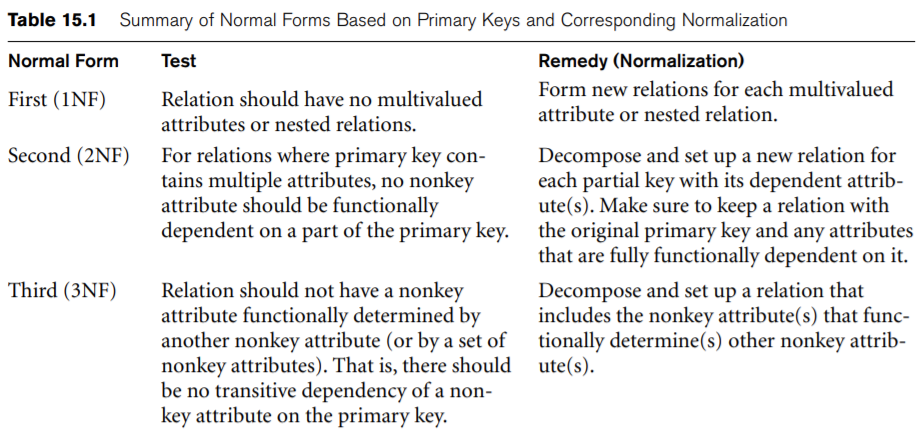
The update of a piece of information must occur at multiple locations (if not, it leads to inconsistency).

### Deletion Anomalies

The deletion of a piece of information unintentionally removes other information.

# 

# Normal Forms



## First Normal Form (1NF)

* All attributes dependent on primary key
* All rows should contain atomic values. Keep it boiled down to a single value
* No mix-and-match types within a given column
* Columns should have unique names
* Order doesn’t matter within rows

A table that contains one or more repeating groups is not a relation by definition. And it is not in the 1st normal form

### Converting to 1NF

1. Identify the key
2. Detect anomalies
3. Decompose into smaller relations
4. Prove that the decomposition is good!

## Second Normal Form (2NF)

A relation R is in 2NF if it is in 1NF and every non prime-key attribute is fully functionally dependent on the primary key.

A **non-prime-key** attribute is one that is not a part of the primary key.

For a given table, it is in 2NF if:

* It is in at least 1NF
* Should not have any partial dependencies

Can we form a like-primary key using other columns?

## Third Normal Form (3NF)

A relation is in 3NF if it is in 2NF and in which every non-prime-key attribute is non-transitively dependent on the primary key.

For a given table, it is in 2NF if:

* If it is in 2NF
* There are no transitive dependencies

If there are a bunch of transitive dependencies, then the lookup time for a given value would be very significant if the value was far down the dependency list.

## Boyce-Codd Normal Form (BCNF)

For a given table, it is in BCNF if:

* It is in 3NF
* For any dependency , should be a super key

The table has to be in 3NF and for any dependency A -> B, A should be a super key. In other words, if A is a non-prime attribute, B cannot be a prime attribute. In order to make a table in BCNF, The A attribute must be broken off into a separate table linked by its primary key to the original table.

## 

## Fourth Normal Form (4NF)

Aim is to remove any multivalued dependency. A multivalued dependency is defined as such: With A -> B for a single value of A, more than one value of B exists. This can be solved by splitting each value of B into its own table.

# Functional Dependency

Given two attributes X and Y of a relation R, Y is functionally dependent on X if whenever two tuples agree on their X-value, they also agree on their Y-value (each X-value in R has associated with it exactly one Y-value in R).

During normalization, the functional dependencies of interest are non-trivial and ones that hold for all times.

To find the set of all FDs, we can either use the requirements given (method commonly used in real-world scenarios) OR use an algorithm that finds the closure of X under F, where X is a set of attributes and F is a set of FDs.

The set of all attributes functionally determined by X under F is called the closure of X under F and denoted by X+

## Syntax

means “X functionally determines Y” or “Y is functionally dependent on X”

X is the determinant of the functional dependency , and Y is the dependent.

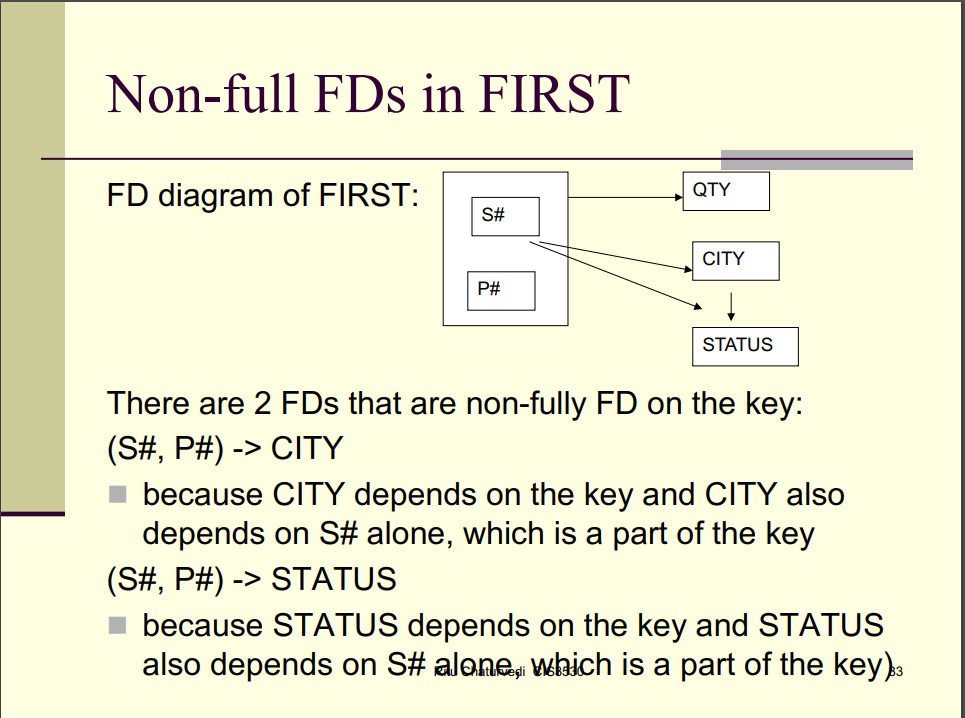
## Properties

* X may or may not be the key attribute of R n X and Y may be composite
* X and Y could be mutually dependent on each other
* The same Y-value may occur in multiple tuples
* May be **trivial** or **non-trivial**

## Full Functional Dependency

Given attributes X and Y of a relation R, Y is fully functionally dependent on X if Y is functionally dependent on X but not on any proper subset of X.

## Example of Non-Full FDs



## Transitive Dependency

Given attributes X, Y and Z of a relation R, Z is transitively dependent on X if

## Non-Loss Decomposition

A bad decomposition loses information.

It is **impossible** to examine all the instances of a relation schema for testing if a decomposition is lossless.

In a good decomposition:

* The join of decomposed relations restores the original relation.
* Decomposed relations can be maintained independently

### Rissanen's Rules

Decomposition of R into R1 and R2 is good if n The common attributes of R1 and R2 form a candidate key for at least one of the pair AND n Every FD in R can be logically derived from those in R1 and R2

### 

### CREATE TABLE

Creates a new relation by specifying the name, as well as the attributes with their data types.

* Data Types
  + INTEGER, INT
  + REAL, DECIMAL, NUMERIC
  + Fixed length: CHAR(n)
  + Varying length: VARCHAR(n)

### Defining Primary Key

A primary key is defined by adding PRIMARY KEY to the end of the primary key attribute

### Defining Composite Key

A composite key is defined by adding an element at the end PRIMARY KEY(A1, A2)

### Defining Foreign Key

A foreign key is defined by adding an element at the end FOREIGN KEY (A1) REFERENCES RELATION

### Defining A Constraint

A constraint can be added by adding CHECK(CONDITION) to the end of an attribute, (example) ATTR1 NUMERIC CHECK (ATTR1 > 0)

### Creating a Table from another Table

CREATE TABLE TABLECPY AS SELECT \* FROM TABLE1

By adding WHERE (condition that is always false), only the structure of TABLE1 is copied.

### ALTER TABLE

* Used to add an attribute to one of the base relations
* Syntax: ALTER TABLE tablename (operation)
* Operations
  + ADD
    - Ex. ALTER TABLE Test ADD b INTEGER;
  + ALTER
    - Ex. ALTER TABLE Test ALTER b TYPE TEXT;
  + DROP
    - Ex. ALTER TABLE S DROP d;
  + ADD CONSTRAINT
    - ALTER TABLE Test ADD CONSTRAINT pk1 PRIMARY KEY (a);

### DROP TABLE

* Removes a relation and its definition.
* Syntax: DROP TABLE DEPARTMENT; or DROP TABLE IF EXISTS DEPARTMENT;
* Adding CASCADE constraints to the end removes any foreign key constraints
  + DROP TABLE S CASCADE CONSTRAINTS

### TRUNCATE TABLE

Similar to DELETE where it removes the rows from a database. However, it does not scan the database and mark rows for deletion like DELETE does. Instead TRUNCATE deletes the rows right away making it faster than DELETE. This means the rows removed through TRUNCATE cannot be rolled back.

Insertion anomalies

Functional dependencies

Is X in 1NF/2NF?

Decompose X into smaller relations

Prove that a decomposition is non-loss

Is X in BCNF?

CREATE\_TABLE()

CREATE\_TABLE using another table

Is X a serializable schedule?

What are the equivalent serial schedules?

# Precedence Graphs

If a graph contains a cycle then it is not serializable.

## Drawing

* Start from the lowest order node that has a read.
* Draw a line from that node to any it has an associated write with.
  + To have an associated write, it must be writing the **same resource** as the previous read.
* Go to the next node.

Definitions

# Prime Attribute

Prime attributes are the attributes of the candidate key which defines the uniqueness

Determinants of an FD Diagram

# Conflict Definitions

## Phantom Read

A phantom read occurs when, in the course of a transaction, two identical queries are executed, and the collection of rows returned by the second query is different from the first.

## Dirty Read/Write-Read

This occurs when a transaction is allowed to read a row that has been modified by another transaction which is not committed yet at the time the dirty read occurred.

## Lost Update

# (Un/Non)repeatable Read

A non-repeatable read occurs, when during the course of a transaction, a row is retrieved twice and the values within the row differ between reads.

## UNDO Logs

# Non-Serial Schedule

This occurs when two transactions are executing at the same time.