# Spatial Database Management
# GEP 664 / GEP 380
## Class #4: Database design

Frank Donnelly

Dept of EEGS, Lehman College CUNY

Spring 2019

## Today's Topics



## Design Aspects

A good database design should be able to:

- Solve the initial problem / fulfill original mission
- Hold the required data
- Support necessary relationships
- Maintain data integrity
- Maintain data efficiency
- Accommodate future change

## Database Design Stages

Most texts on database design focus on these stages, and presume a database is supporting multiple, long-term applications for a variety of users.

1. Requirements analysis
2. Conceptual model and logical design
3. Physical design

But there is a distinction between designing with this premise versus other use cases: a means to an end for analysis, or a warehouse for long-term storage.

## Requirements analysis

Summarize what the objective of the database is in a few sentences

- Background research
- Organizational culture
- Interview stakeholders
- No technical jargon or details
- Existing database? Study it

## Conceptual Model and Logical design

Analyze the needs and requirements of data creators and users and describe them using models and diagrams

- Identify entities and attributes
- Determine relationships between entities
- Diagram or model
- Validate the design

## Entity Relationship Model

Conceptual model that transforms database requirements into a formal description of entities that appear in a database.

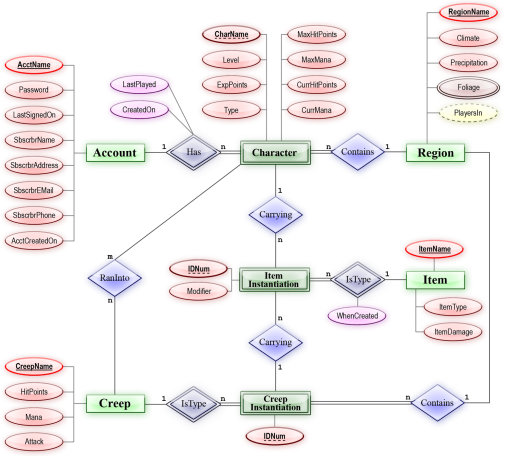



Image source: https://en.wikipedia.org/wiki/Entity-relationship_model

## Chen's ER Model Notation

**Entity** primary data objects. Can be subdivided into individual instances and common types

**Attribute** identifiers or descriptors of entities (and sometimes of relationships)

**Relationship** an association between or among entities

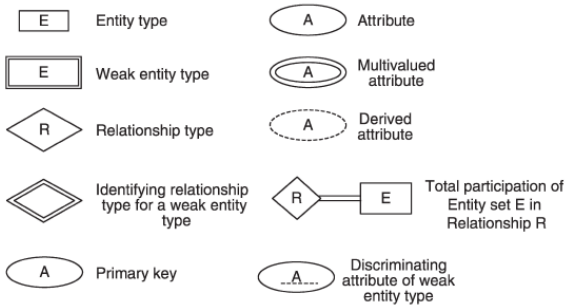



Image source: Song & Chen, Encyclopedia of Database Systems, 2009

## Cardinality

A constraint that specifies the maximum number of entities that can be associated with another entity via a relationship.

One to one (1:1) One customer can have at most one account. One account cannot be owned by more than one customer


One to many (1:N) One customer can have many accounts. One account cannot be owned by more than one customer


Many to many (N:M) One customer can have many accounts. One account may be owned by many customers


## Participation

A constraint that specifies the minimum number of entities that can be associated with another entity via a relationship.

Total (mandatory) When every instance of an entity must participate in a given relationship, i.e. all customers must have at least one account. Indicated by double relationship lines.

Partial (optional) When every instance of an entity is not required to participate in a given relationship, i.e. not all customers are required to have an account. Indicated by single relationship lines.

## Chen ER Model Example

Relationships between entities can be binary (between 2), ternary (between 3), or recursive (within 1).

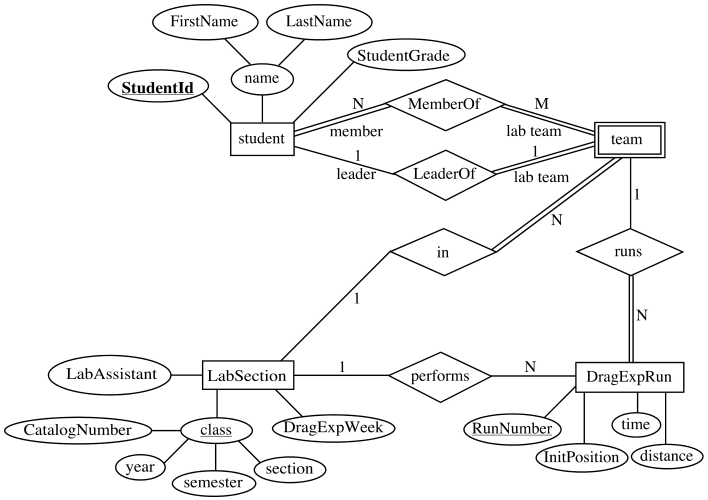



Image source: https://wofford-ecs.org/DataAndVisualization/ermodel/

## Many Modeling Notations

There are several different modeling notations... Chen is on the left, UML is on the right

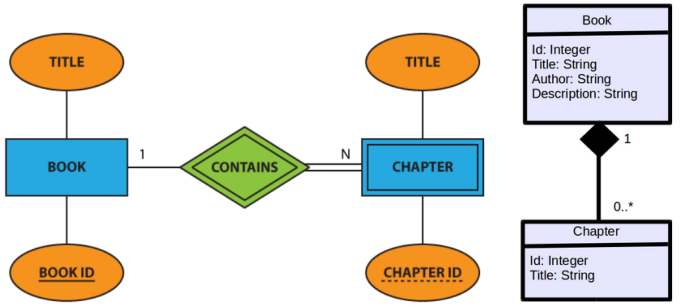



Image source: http://www.vertabelo.com/blog/technical-articles/chen-erd-notation

## Physical design

Translate the logical design into a physical model that is implemented with specific hardware and software

- Finalize keys
- Determine names for attributes
- Determine data types
- Create constraints
- Create indexes
- Fine-tune database for efficiency

## Keys

Moving from the logical to the physical design, identify candidate keys; attributes that can serve as identifiers.

- Primary Key insures entity integrity
  - Unique, not null id for each row
  - Natural key is derived outside the database, has meaning
  - Surrogate key is artificial, made in the database
  - Composite key consists of more than one column
- Foreign Key insures referential integrity
  - A primary key in one table can be a foreign key in another
  - Links row in child table to a parent table
  - Prevents inconsistencies between tables

## Standard Codes

Certain entities may be identified using a standard system of codes. If your entities have a standard you should use it, and if your data lacks this attribute you should assign it.

ISO codes for countries

| English short name | French short name | Alpha-2 code | Alpha-3 code | Numeric |
|---|---|---|---|---|
| Afghanistan | Afghanistan (l') | AF | AFG | 004 |
| Åland Islands | Åland(les Îles) | AX | ALA | 248 |
| Albania | Albanie (l') | AL | ALB | 008 |
| Algeria | Algérie (l') | DZ | DZA | 012 |
| American Samoa | Samoa américaines (les) | AS | ASM | 016 |

Image source: https://www.iso.org/obp/ui/

ANSI FIPS codes for US geography

| Name | FIPS State Numeric Code | Official USPS Code |
|---|---|---|
| Alabama | 01 | AL |
| Alaska | 02 | AK |
| Arizona | 04 | AZ |
| Arkansas | 05 | AR |
| California | 06 | CA |

Image source: https://www.census.gov/geo/reference/ansi_statetables.html

## Domains

- Generally refers to an attributes allowable set of values
- Specifically, another way of stating constraints that can be applied to multiple objects

```
CREATE DOMAIN nyc.wcolor AS VARCHAR(20)
  DEFAULT 'BLUE'
  CHECK (VALUE IN ('BLUE', 'RED', 'YELLOW', 'GREEN'));
```

```
  CREATE TABLE nyc.weather_alert (
  aid serial PRIMARY KEY,
  aname nyc.wcolor NOT NULL,
  conditions text
  );
```

## Indexes

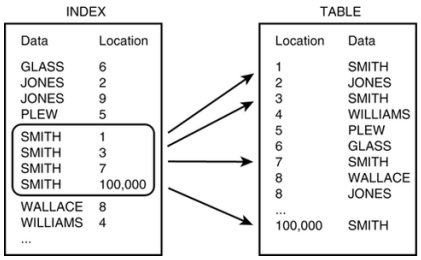



Image source:
https://en.wikipedia.org/wiki/Database_index

- Speeds up access based on a specific column
- Keys are indexed by default
- Values can be unique or not (unique is best)
- Bad for small tables or small set of values

```
CREATE INDEX indexname
ON tablename (column);
```

## ER Exercise - GIS Lab Workshops

Create ER Diagram to model GIS workshop program

## Today's Topics

Database Design

Normalization

Next Class

## Normal Forms

Rules for organizing attributes and tables to reduce data redundancy and improve data integrity

- Part of Codd's original paper on the relational model
- Databases that are not well normalized are harder to maintain and are error prone
- There are 5 normal forms; the first 3 are common
- Sometimes necessary to sacrifice normal form in favor of usability

## First Normal Form

- The order of the rows must be irrelevant; they cannot have an inherent order
- There can be no duplicate records; every record must be unique and have a primary key
- For every record, a column must contain only one logical value that cannot be subdivided; no lists or arrays
- In some interpretations, records within a table should represent the same atomic, indivisible elements; no totals or subtotals

| weather_id integer | station_id character varying(20) | thedate timestamp without time zone | skycondition text | drybulb_temp_f integer |
|---|---|---|---|---|
| 121846 | WBAN:94728 | 2013-12-01 00:51:00 | BKN:07 21 OVC:08 47 | 39 |
| 121847 | WBAN:94728 | 2013-12-01 01:20:00 | FEW:02 20 OVC:08 49 | 39 |
| 121848 | WBAN:94728 | 2013-12-01 01:51:00 | BKN:07 50 | 37 |
| 121849 | WBAN:94728 | 2013-12-01 02:51:00 | FEW:02 25 BKN:07 38 OVC:08 46 | 36 |

skyconditions violates 1NF as it's a list that can be subdivided

## Second Normal Form

Every non-key column must be relevant or dependent on the primary key. Attributes that are not should be divided into separate tables.

| weather_id integer | station_id character varying(20) | station_name text | elevation numeric(6,1) | lat numeric(9,6) | lon numeric(9,6) | thedate timestamp without time zone | skycondition text | drybulb_temp_f integer |
|---|---|---|---|---|---|---|---|---|
| 121846 | WBAN:94728 | NEW YORK CENTRAL PARK | 39.6 | 40.778890 | -73.969170 | 2013-12-01 00:51:00 | BKN:07 21 | 39 |
| 121847 | WBAN:94728 | NEW YORK CENTRAL PARK | 39.6 | 40.778890 | -73.969170 | 2013-12-01 01:20:00 | FEW:02 20 | 39 |
| 121848 | WBAN:94728 | NEW YORK CENTRAL PARK | 39.6 | 40.778890 | -73.969170 | 2013-12-01 01:51:00 | BKN:07 50 | 37 |
| 121849 | WBAN:94728 | NEW YORK CENTRAL PARK | 39.6 | 40.778890 | -73.969170 | 2013-12-01 02:51:00 | FEW:02 25 | 36 |

elevation, lat, and long violate 2NF as they do not depend on the primary key

Question: Does each column describe what the primary key identifies?

## Third Normal Form

Every non-key column must be relevant or dependent on the primary key and not on any other column; there should be no transitive dependencies (where A depends on B, and B depends on C, and therefore A depends on C). Attributes that are transitive should be divided into separate tables.

| weather_id integer | station_id character varying(20) | station_name text | thedate timestamp without time zone | skycondition text | drybulb_temp_f integer |
|---|---|---|---|---|---|
| 121846 | WBAN:94728 | NEW YORK CENTRAL PARK | 2013-12-01 00:51:00 | BKN:07 21 | 39 |
| 121847 | WBAN:94728 | NEW YORK CENTRAL PARK | 2013-12-01 01:20:00 | FEW:02 20 | 39 |
| 121848 | WBAN:94728 | NEW YORK CENTRAL PARK | 2013-12-01 01:51:00 | BKN:07 50 | 37 |
| 121849 | WBAN:94728 | NEW YORK CENTRAL PARK | 2013-12-01 02:51:00 | FEW:02 25 | 36 |

station_name violates 3NF as it depends on weather_id and station_id

Question: Does each column depend only on the primary key? Or are there other dependecies?

## Transitive Dependencies

Which columns are transitive?

| Primary Key | Column A | Column B |
|---|---|---|
| ObservationID | Temperature | SkyCondition |
| ObservationID | Temperature | AboveFreezing |
| PersonID | Age | Birthdate |
| PersonID | Age | LastName |
| CountryID | ContinentID | ContinentName |
| CountryID | CountryName | Population |
| ProductID | Model | Manufacturer |
| ProductID | Model | Color |

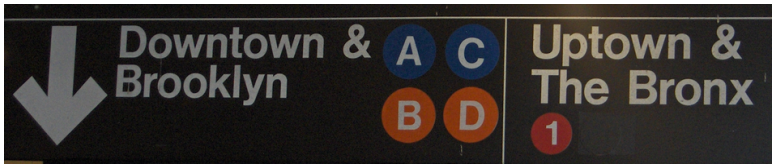
## Example - NYC Subway

In this example we'll denormalize data on NYC subway services:

- **service** a specific train line
- **division** historical groupings of trunk lines and services
- **trunk_line** shared tracks used by services in midtown Manhattan
- **color** assigned to services that share a trunk line
- **service_type** indicates whether the service makes express or local stops in Manhattan
- **services** the number of services that run along a trunk

## 0NF

| train | color | service_type | services |
|---|---|---|---|
| A IND 8th Ave Line | blue | express | 3 |
| B IND 6th Ave Line | orange | local | 4 |
| C IND 8th Ave Line | blue | local | 3 |
| D IND 6th Ave Line | orange | express | 4 |
| 1 IRT Bway - 7th Ave Line | red | local | 3 |
| A IND 8th Ave Line | blue | express | 3 |




## 1NF

| service | division | trunk_line | color | service_type | services |
|---|---|---|---|---|---|
| A | IND | 8th Ave Line | blue | express | 3 |
| B | IND | 6th Ave Line | orange | local | 4 |
| C | IND | 8th Ave Line | blue | local | 3 |
| D | IND | 6th Ave Line | orange | express | 4 |
| 1 | IRT | Bway - 7th Ave Line | red | local | 3 |

Removed duplicates, split list of values in trains column into multiple columns. Service becomes primary key.

## 2NF

| service | division | trunk_line | color | service_type |
|---|---|---|---|---|
| A | IND | 8th Ave Line | blue | express |
| B | IND | 6th Ave Line | orange | local |
| C | IND | 8th Ave Line | blue | local |
| D | IND | 6th Ave Line | orange | express |
| 1 | IRT | Bway - 7th Ave Line | red | local |

| trunk_line | services |
|---|---|
| 6th Ave Line | 4 |
| 8th Ave Line | 3 |
| Bway - 7th Ave Line | 3 |

Number of services does not describe or depend on service primary key; it depends on trunk_line, so break that out into separate table for lines and remove duplicates.

## 3NF

| service | trunk_line | service_type |
|---|---|---|
| A | 8th Ave Line | express |
| B | 6th Ave Line | local |
| C | 8th Ave Line | local |
| D | 6th Ave Line | express |
| 1 | Bway - 7th Ave Line | local |

| trunk_line | division | color | services |
|---|---|---|---|
| 6th Ave Line | IND | orange | 4 |
| 8th Ave Line | IND | blue | 3 |
| Bway - 7th Ave Line | IRT | red | 3 |

Division and color have transitive dependencies; they depend on service and on trunk_line. Move them to the line table.

## Handling Lists of Attributes

Avoid grouping values into single columns

| sid | sname | platforms | trains |
|---|---|---|---|
| mn056 | 50th St | 4 | C E |
| mn059 | 59th St Columbus Circle | 5 | A B C D 1 |
| mn064 | 72nd St | 2 | B C |

Avoid creating indeterminate columns with many nulls

| sid | sname | platforms | train1 | train2 | train3 | train4 | train5 |
|---|---|---|---|---|---|---|---|
| mn056 | 50th St | 4 | C | E | | | |
| mn059 | 59th St Columbus Circle | 5 | A | B | C | D | 1 |
| mn064 | 72nd St | 2 | B | C | | | |

A limited number of attributes can become booleans

| sid | sname | platforms | train_a | train_b | train_c | train_d | train_e | train_1 |
|---|---|---|---|---|---|---|---|---|
| mn056 | 50th St | 4 | no | no | yes | no | yes | no |
| mn059 | 59th St Columbus Circle | 5 | yes | yes | yes | yes | no | yes |
| mn064 | 72nd St | 2 | no | yes | yes | no | no | no |

## Break Tables Apart...

One train can stop at many stations, and one station can host many trains. For many to many relationships, it is best practice to break the tables apart.

| sid | sname | platforms |
|---|---|---|
| mn056 | 50th St | 4 |
| mn059 | 59th St Columbus Circle | 5 |
| mn064 | 72nd St | 2 |

| train | service |
|---|---|
| A | express |
| B | local |
| C | local |
| D | express |
| E | local |
| 1 | local |

## ...Use Bridge Tables

Bridge tables are used for handling many to many relationships. They include the primary key of each table so the two can be related.

| sid | train |
|---|---|
| mn056 | C |
| mn056 | E |
| mn059 | A |
| mn059 | B |
| mn059 | C |
| mn059 | D |
| mn059 | 1 |
| mn064 | B |
| mn064 | C |

| sidt | sid | train |
|---|---|---|
| mn056-C | mn056 | C |
| mn056-E | mn056 | E |
| mn059-A | mn059 | A |
| mn059-B | mn059 | B |
| mn059-C | mn059 | C |
| mn059-D | mn059 | D |
| mn059-1 | mn059 | 1 |
| mn064-B | mn064 | B |
| mn064-C | mn064 | C |

Primary key of bridge table is either a composite key or a new key with concatenated values.

## Breaking the Rules

The rules of normalization may occasionally be broken for the sake of usability; i.e. the user can access values more readily without doing multiple joins.

Within the realm of GIS, it may be necessary to break the rules of normalization in some instances:

- Single columns with groups of values may be needed for map labels
- GIS software cannot accommodate composite keys
- Many to many relationships are difficult to work with in GIS

## Limits of Database Design Models

### In the Field of Geography

The tight relationships that are modeled in database design may not always exist. The primary relationship between different entities may be that they exist in close proximity in geographic space.

### Data Warehousing

In this general model, large datasets are stored together to take advantage of the efficiencies that relational databases provide. The entities themselves may be loosely related.

## Today's Topics

Database Design

Normalization

Next Class

## Due Next Class

The following are due at the beginning of our next class:

### Assignment #4

Posted on the course website

### Readings for Class #5

Listed in the syllabus, in the *Practical SQL* book