COMP3311 24T1 Database Systems

week 7 - 2

Outline

- **Announcements**
- Assignment 2
- Recap
- Relational Design Theory & Redundancy
- Functional Dependencies

Announcement 1

- **Assignment - 2**
  - specifications will be released later this week
  - due date: Week 10 Wednesday (17th April) 10:00pm
  - practicing pyscopg2 (actually still mainly about SQL queries)
  - will be using Python

- **Do it earlier!**

- **Use private posts to post your assignment code!**

Announcement 2

- **Quiz 4**
  - Due on this Friday (29 Mar)
  - Just do it!

Announcement 3

- **Help Session**
  - Thursday session to be shifted to 11:00 - 13:00 (G05)

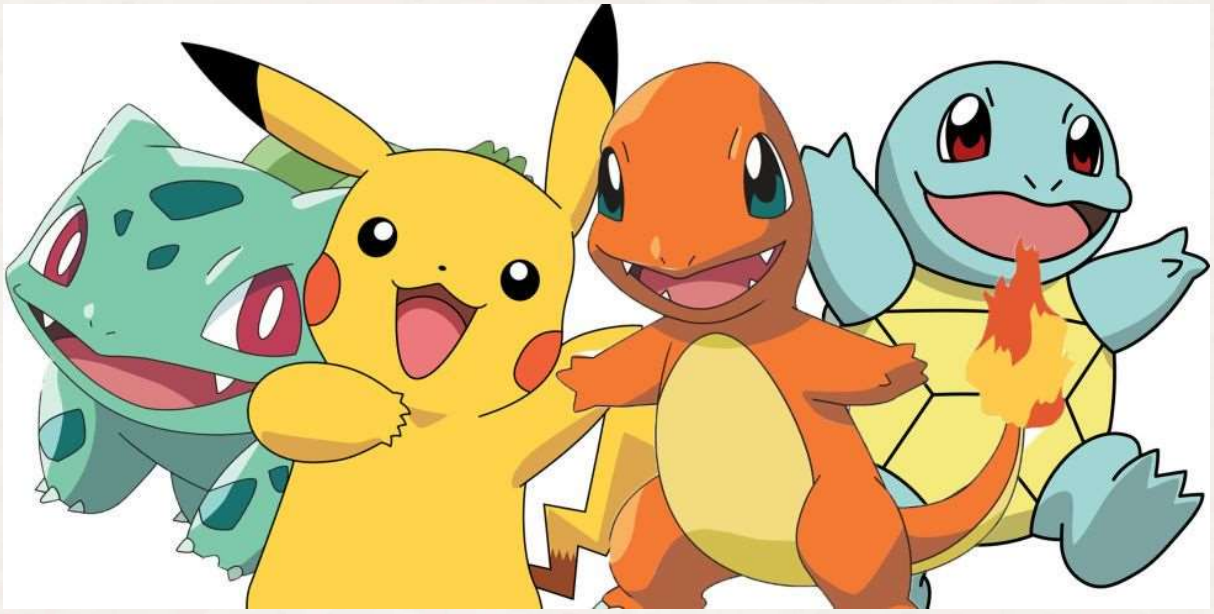
Outline

- Announcements
- **Assignment 2**
- Recap
- Relational Design Theory & Redundancy
- Functional Dependencies

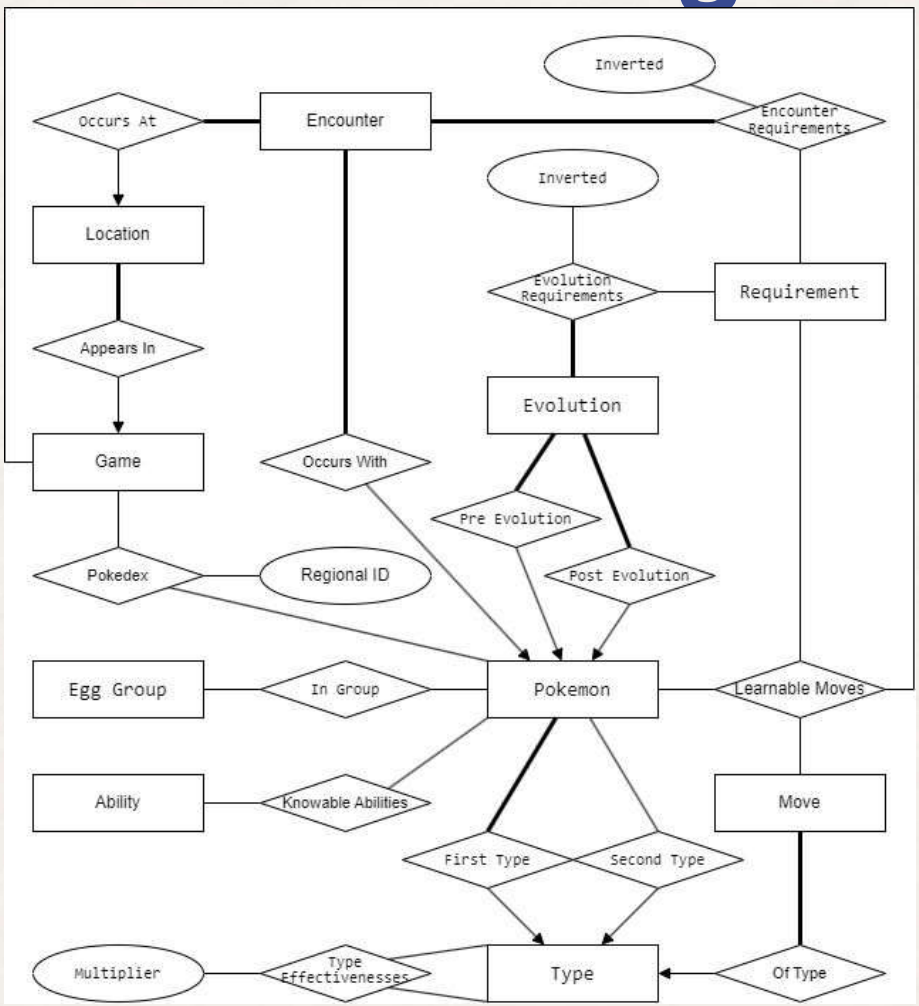
Assignment 2 - Database

Which db will we use here?

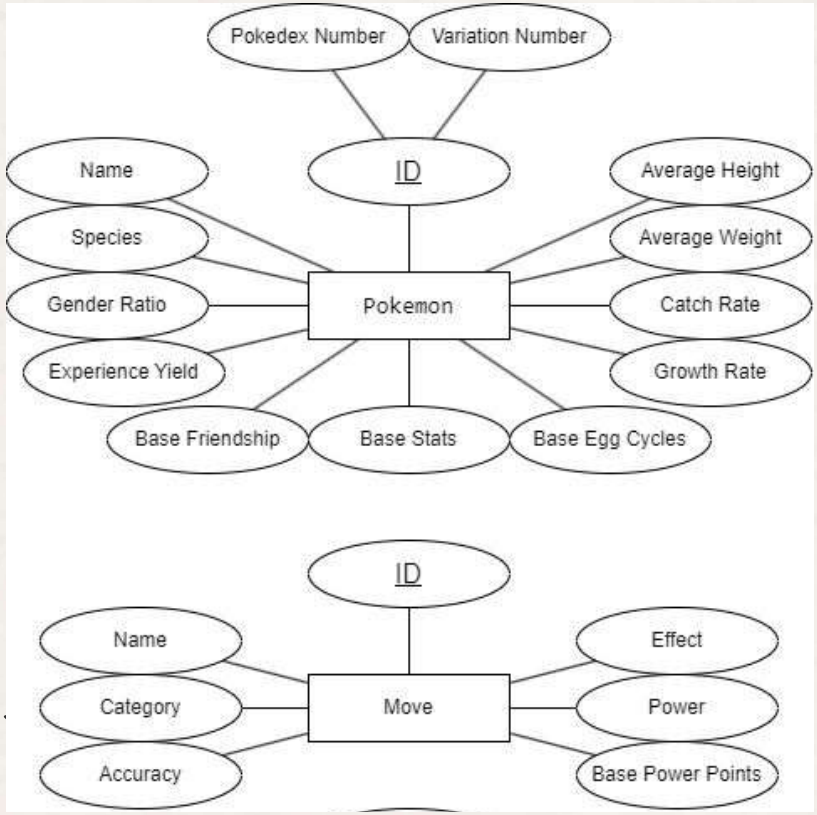
An assignment that was complained as being “too hard” ... 😱

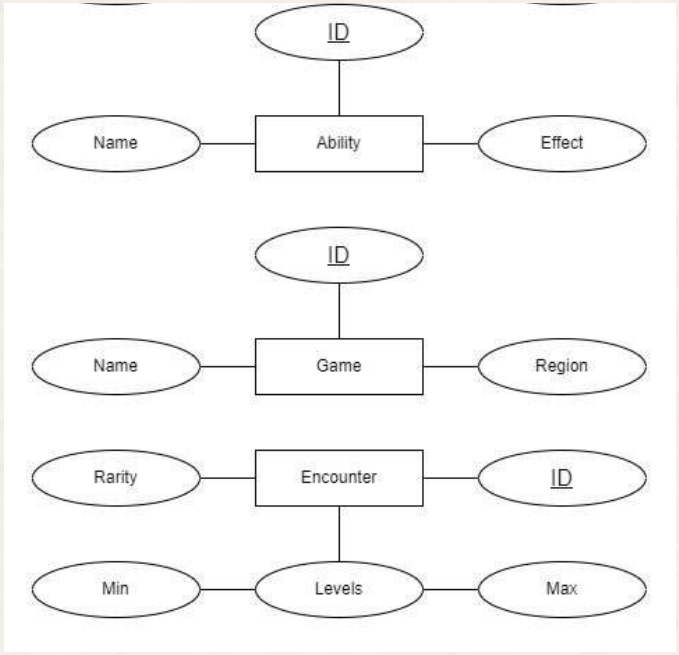


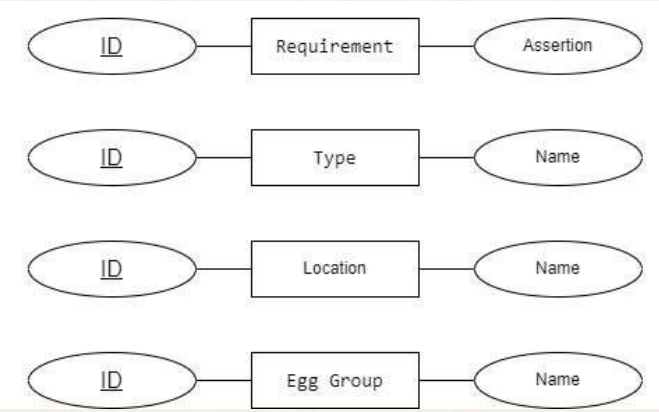
Assignment 2 - ER Diagram



Assignment 2 – ER Diagram







Assignment 2 - DB Setup

Assignment 2 Schema & Data dump:

https://webcms3.cse.unsw.edu.au/COMP3311/24T1/resources/97244

- **setup the DB earlier!**

Assignment 2 – Why it was hard?

A person who can speak multiple languages is called a ...

- **polyglot**!

Programs involving multiple programming languages are also called **polyglot** programs!

Need to be careful when mapping data types across languages!

- especially for floating point numbers!

⚠️Try to do data processing as much as possible on the SQL side!

Outline

- Announcements
- Assignment 2
- **Recap**
- Relational Design Theory & Redundancy
- Functional Dependencies

Recap

Python

- basic syntax, expressions, data types, definition of function, command line arguments

Psycopg2 (psy|co|p|g|2)

- import, connect, cursor (execute, fetch - 3, invoke pgsql functions)

Outline

- Announcements
- Assignment 2
- Recap
- **Relational Design Theory & Redundancy**
- Functional Dependencies

Relational Design Theory

The aim of studying relational design theory:

- improve understanding of relationships among data
- gain enough formalism to assist practical database design

What we study here:

- basic theory and definition of **functional dependencies**
- methodology for improving schema designs (**normalisation**)

Functional dependencies:

- describe relationships between **attributes within a relation**
- have implications for "good" relational schema design

Relational Design and Redundancy

A **good** relational database design:

- must capture **all** necessary attributes/associations
- do this with **minimal** amount of stored information

Minimal stored information ⇒ no redundant data.

In database design, redundancy is **generally** a "bad thing":

- causes problems maintaining consistency after updates

But ... redundancy may give performance improvements

- e.g. avoid a join to collect pieces of data together

E

Redundancy Example

Consider the following relation defining bank accounts/branches:

- asset ⇒ the total amount of money stored in a branch

accountNo	balance	customer	branch	address	assets
A-101	500	1313131	Downtown	Brooklyn	9000000
A-102	400	1313131	Perryridge	Horseneck	1700000
A-113	600	9876543	Round Hill	Horseneck	8000000
A-201	900	9876543	Brighton	Brooklyn	7100000
A-215	700	1111111	Mianus	Horseneck	400000
A-222	700	1111111	Redwood	Palo Alto	2100000
A-305	350	1234567	Round Hill	Horseneck	8000000
...	...	...	...	...	...

Careless updating of this data may introduce inconsistencies.

E

Redundancy Example

If we add $300 to account A-113 ...

accountNo	balance	customer	branch	address	assets
A-101	500	1313131	Downtown	Brooklyn	9000000
A-102	400	1313131	Perryridge	Horseneck	1700000
A-113	900	9876543	Round Hill	Horseneck	8000300
A-201	900	9876543	Brighton	Brooklyn	7100000
A-215	700	1111111	Mianus	Horseneck	400000
A-222	700	1111111	Redwood	Palo Alto	2100000
A-305	350	1234567	Round Hill	Horseneck	8000000
...	...	...	...	...	...

E

Redundancy Example

If we add a new account A-306 at the Round Hill branch ...

accountNo	balance	customer	branch	address	assets
A-101	500	1313131	Downtown	Brooklyn	9000000
A-102	400	1313131	Perryridge	Horseneck	1700000
A-113	900	9876543	Round Hill	Horseneck	8000300
A-201	900	9876543	Brighton	Brooklyn	7100000
A-215	700	1111111	Mianus	Horseneck	400000
A-222	700	1111111	Redwood	Palo Alto	2100000
A-305	350	1234567	Round Hill	Horseneck	8000000
A-306	500	7654321	Round Hill	Horseneck	8000500?
...	...	...	...	...	...

E

Redundancy Example

If we close account A-101 ...

accountNo	balance	customer	branch	address	assets
~~A-101~~	~~500~~	~~1313131~~	~~Downtown~~	~~Brooklyn~~	~~9000000~~
A-102	400	1313131	Perryridge	Horseneck	1700000
A-113	900	9876543	Round Hill	Horseneck	8000300
A-201	900	9876543	Brighton	Brooklyn	7100000
A-215	700	1111111	Mianus	Horseneck	400000
A-222	700	1111111	Redwood	Palo Alto	2100000
A-305	350	1234567	Round Hill	Horseneck	8000000
A-306	500	7654321	Round Hill	Horseneck	8000500?
...	...	...	...	...	...

Redundancy – anomalies

Insertion anomaly:

- when we insert a new record, we need to check that branch data is consistent with existing tuples

Update anomaly:

- e.g. if a branch changes address, we need to update all tuples referring to that branch

Deletion anomaly:

- e.g. if we remove information about the last account at a branch, all of the branch information disappears

Insertion/update anomalies can be handled, e.g. by triggers

- but this requires extra DBMS work on every change to the database

Database Design (revisited)

To avoid these kinds of update problems:

- need a schema with "**minimal overlap**" between tables
- each table contains a "**coherent**" collection of data values

Such schemas have little/no redundancy

ER → SQL mapping tends to give non-redundant schemas

- but does not guarantee no redundancy

Outline

- Announcements
- Assignment 2
- Recap
- Relational Design Theory & Redundancy
- **Functional Dependencies**

Notation/Terminology

Most texts adopt the following terminology:

- **Attributes** upper-case letters from start of alphabet (e.g. A, B, C, ...)
- **Sets of attributes** concatenation of attribute names (e.g. X=ABCD, Y=EFG)
- **Relation schemas** upper-case letters, denoting set of all attributes (e.g. R)
- **Relation instances** lower-case letter corresponding to schema (e.g. r(R))
- **Tuples** lower-case letters (e.g. t, t', t1, u, v)
- **Attributes in tuples** tuple[attrSet] (e.g. t[ABCD], t[X])

Functional Dependency

A relation instance r(R) satisfies a dependency $X \rightarrow Y$ if

- for any t, u $\in$ r, $t[X] = u[X] \Rightarrow t[Y] = u[Y]$

In other words, if two tuples in R agree in their values for the set of attributes X, then they must also agree in their values for the set of attributes Y.

- **Attributes** (e.g. A, B, C, ...)
- **Sets of attributes** (e.g. X=ABCD, Y=EFG)
- **Relation schemas** (e.g. R)
- **Relation instances** (e.g. r(R))
- **Tuples** (e.g. t, t', t1, u, v)
- **Attributes in tuples** (e.g. t[ABCD], t[X])

We say that "Y is **functionally dependent** on X ".
Attribute sets X and Y may overlap; it is trivially true that $X \rightarrow X$.

Notes:

- $X \rightarrow Y$ can also be read as "X determines Y"
- the single arrow $\rightarrow$ denotes functional dependency
- the double arrow $\Rightarrow$ denotes logical implication

E

Functional Dependency Example 1

Consider the following (redundancy-laden) relation:

Title	Year	Len	Studio	Star
King Kong	1933	100	RKO	Fay Wray
King Kong	1976	134	Paramount	Jessica Lange
King Kong	1976	134	Paramount	Jeff Bridges
Mighty Ducks	1991	104	Disney	Emilio Estevez
Wayne's World	1995	95	Paramount	Dana Carvey
Wayne's World	1995	95	Paramount	Mike Meyers

Consider the following (redundancy-laden) relation:

Title Year → Studio Len ✔️

Is this a functional dependency? Title Year → Star ❌

E

Functional Dependency Example 2

Consider an instance r(R) of the relation schema R(ABCDE):

A	B	C	D	E
a1	b1	c1	d1	e1
a2	b1	c2	d2	e1
a3	b2	c1	d1	e1
a4	b2	c2	d2	e1
a5	b3	c3	d1	e1

Since A values are unique, the definition of fd gives:

- $A \rightarrow B$, $A \rightarrow C$, $A \rightarrow D$, $A \rightarrow E$, or $A \rightarrow BCDE$

Since all E values are the same, it follows that:

- $A \rightarrow E$, $B \rightarrow E$, $C \rightarrow E$, $D \rightarrow E$?

E

Functional Dependency Example 2 (cont)

Consider an instance r(R) of the relation schema R(ABCDE):

A	B	C	D	E
a1	b1	c1	d1	e1
a2	b1	c2	d2	e1
a3	b2	c1	d1	e1
a4	b2	c2	d2	e1
a5	b3	c3	d1	e1

In practice, choose a minimal set of fds (**basis**):

- from which all other fds can be derived
- which captures useful problem-domain information

Other observations:

- combinations of BC are unique, therefore BC → ADE
- combinations of BD are unique, therefore BD → ACE
- if C values match, so do D values, therefore C → D
- however, D values don't determine C values, so !(D → C)

We could derive many other dependencies, e.g. AE → BC, ...

Functional Dependency

Above examples consider dependency within a relation instance r(R).

More important for design is dependency across all possible instances of the relation (i.e. a schema-based dependency).

This is a simple generalisation of the previous definition:

- for any t, u ∈ **any** r(R), t[X] = u[X] ⇒ t[Y] = u[Y]

- **Attributes** (e.g. A, B, C, ...)
- **Sets of attributes** (e.g. X=ABCD, Y=EFG)
- **Relation schemas** (e.g. R)
- **Relation instances** (e.g. r(R))
- **Tuples** (e.g. t, t', t1, u, v)
- **Attributes in tuples** (e.g. t[ABCD], t[X])

Such dependencies tend to capture semantics of problem domain.

Functional Dependency - Generalization

Can we generalise some ideas about functional dependency?

E.g. are there dependencies that hold for any relation?

- yes, but they're generally trivial, e.g. $Y \subset X \Rightarrow X \rightarrow Y$

E.g. do some dependencies suggest the existence of others?

- yes, rules of inference allow us to derive dependencies
- allow us to reason about sets of functional dependencies

Inference Rules

Armstrong's rules are general rules of inference on fds.

F1. **Reflexivity** e.g. $X \rightarrow X$

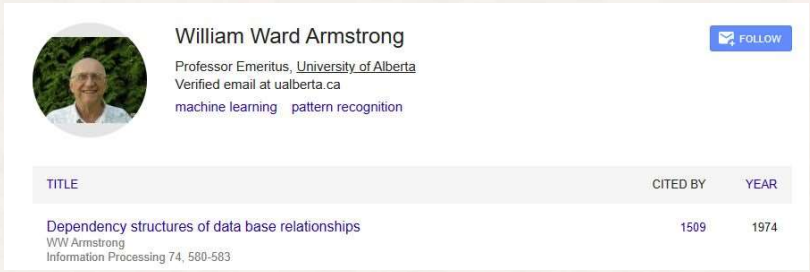
- a formal statement of trivial dependencies; useful for derivations

F2. **Augmentation** e.g. $X \rightarrow Y \Rightarrow XZ \rightarrow YZ$

- if a dependency holds, then we can expand its left hand side (along with RHS)

F3. **Transitivity** e.g. $X \rightarrow Y, Y \rightarrow Z \Rightarrow X \rightarrow Z$

- the "most powerful" inference rule; useful in multi-step derivations



Inference Rules

Some other useful rules exist:

F4. **Additivity** e.g. $X \rightarrow Y, X \rightarrow Z \Rightarrow X \rightarrow YZ$

- useful for constructing new right hand sides of fds (also called **union**)

F5. **Projectivity** e.g. $X \rightarrow YZ \Rightarrow X \rightarrow Y, X \rightarrow Z$

- useful for reducing right hand sides of fds (also called **decomposition**)

F6. **Pseudotransitivity** e.g. $X \rightarrow Y, YZ \rightarrow W \Rightarrow XZ \rightarrow W$

- shorthand for a common transitivity derivation

E

Inference Rules Example

Example: determining validity of AB → GH, given:
R = ABCDEFGHIJ
F = { AB → E, AG → J, BE → I, E → G, GI → H }

Derivation:

1.	AB → E	(given)
2.	E → G	(given)
3.	AB → G	(1,2, transitivity)
4.	AB → AB	(1, reflexivity)
5.	AB → B	(4, projectivity)
6.	AB → BE	(1,5, additivity)

- **F1. Reflexivity** e.g. X → X
- **F2. Augmentation** e.g. X → Y ⇒ XZ → YZ
- **F3. Transitivity** e.g. X → Y, Y → Z ⇒ X → Z
- **F4. Additivity** e.g. X → Y, X → Z ⇒ X → YZ
- **F5. Projectivity** e.g. X → YZ ⇒ X → Y, X → Z
- **F6. Pseudotransitivity** e.g. X → Y, YZ → W ⇒ XZ → W

what about (E → G, GI → H, ?)

EI → H (E → G, GI → H, pseudotransitivity)

A shorthand for transitivity?

from (E → G, GI → H) to EI → H

E → G so EI → GI, augmentation

EI → GI, GI → H (given)
so EI → H, transitivity

Thank you!