Outline

- Mechanisms and definitions to work with FDs:
  - Closures, candidate keys, canonical covers etc…
  - Armstrong axioms
- Decompositions:
  - Loss-less decompositions, Dependency-preserving decompositions
- BCNF:
  - How to achieve a BCNF schema
  - BCNF may not preserve dependencies
    - 3NF: Solves the above problem
- Peewee
- Mechanisms:
  - closures of function dependences
  - closures of attributes
  - extraneous attributes
  - canonical covers

Object-Relational Mappings (ORMs)

- Motivation
  - SQL is low-level
  - mapping SQL operations to objects might be more natural
- Examples
  - Django (python)
  - Hibernate
  - Peewee (python)   Why?
    - Ease of use: simple python API for defining models and queries
    - Lightweight: easy to retrofit on existing schemas
    - Flexibility: can use raw SQL if necessary

Peewee

- Model definition:
  - python classes mapped to database schemas
  - each class is a table, each attribute a column
- Query building:
  - from peewee import *
  - query = User.select().where(User.age > 21)
- Relationships
  - foreign keys, complex joins
- Queries:

- Perform a JOIN to get tweets and associated users

```
query = (Tweet
         .select(Tweet, User)              # Selecting columns models
         .join(User)                       # Joining with User
         .where(User.username == 'john'))  # Filtering by username 'john'
```

- Loop through the results:

```
for tweet in query:
    print(f"{tweet.user.username} tweeted: {tweet.content}")
```


Peewee (assignment 3)

- Turn existing schema into object model:

```
pwiz.py -e postgresql -u root -P flightsskewed > orm.py
```

- Define `runORM(jsonFile)`, called by `SQLTesting.py` and test with:

```
def runORM(jsonFile):
    Customers.delete().where(Customers.name == 'bob').execute()
    Airports.delete().where(Airports.airportid == 'PET').execute()

    bob = Customers(name="bob", customerid='cust1010', birthdate='1960-01-15',
                    frequentflieron='SW')
    bob.save(force_insert=True)

    bwi = Airports(airportid='PET', city='Takoma', name='Pete', total2011=2,
                   total2012=4)
    bwi.save(force_insert=True)

    for port in Airports.select().order_by(Airports.name):
        print (port.name)
```

Peewee notes

- `SQLTesting.py` includes `orm.py` automatically, calls `runORM`:

```
def runORM(jsonFile):
    with open(jsonFile) as f:
        for line in f:
            j = json.loads(line)
            if 'newcustomer' in j:
                nc = j['newcustomer']
                ...

            elif 'flewon' in j:
                ...

    populateNumFlights()


def populateNumFlights():
    # clear table
    Numberofflightstaken.delete().execute()

    # Recreate...
```


Outline

- Mechanisms and definitions to work with FDs:
  - Closures, candidate keys, canonical covers etc…
  - Armstrong axioms
- Decompositions:
  - Loss-less decompositions, Dependency-preserving decompositions
- BCNF:
  - How to achieve a BCNF schema
  - BCNF may not preserve dependencies
    - 3NF: Solves the above problem
- Peewee
- Back to FDs: mechanisms
  - closures of function dependences
  - closures of attributes
  - extraneous attributes
  - canonical covers

1. Closure of Functional Dependencies

- Given a set of functional dependencies, *F*, its ***closure***, *F^+,* is all FDs that are implied by FDs in *F*.
  - *e.g. If* A → B, *and* B → C, *then clearly* A → C

- We can find F^+ by applying Armstrong's Axioms:
  - if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ **(reflexivity)**
  - if $\alpha \rightarrow \beta$, then $\gamma\,\alpha \rightarrow \gamma\,\beta$ **(augmentation)**
  - if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ **(transitivity)**

- These rules are
  - sound (generate only functional dependencies that actually hold)
  - complete (generate all functional dependencies that hold)

Additional rules (not Armstrong's axioms)

- If $\alpha \rightarrow \beta$ *and* $\alpha \rightarrow \gamma$, then $\alpha \rightarrow \beta\,\gamma$ **(union)**
- If $\alpha \rightarrow \beta\,\gamma$, then $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$ **(decomposition)**
- If $\alpha \rightarrow \beta$ and $\gamma\,\beta \rightarrow \delta$, then $\alpha\,\gamma \rightarrow \delta$ **(pseudotransitivity)**

- The above rules can be inferred from Armstrong's axioms.

Example (only Armstrong's axioms)

- $R = (A, B, C, G, H, I)$
  $F = \{ A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- Some members of F^+
  - $A \rightarrow H$
    - by transitivity from $A \rightarrow B$ *and* $B \rightarrow H$
  - $AG \rightarrow I$
    - by augmenting $A \rightarrow C$ with G, to get $\mathbf{AG \rightarrow CG}$
    - and then transitivity with $AG \rightarrow \mathbf{CG \rightarrow I}$
  - $CG \rightarrow HI$
    - by augmenting $CG \rightarrow I$ to infer $\mathbf{CG \rightarrow CGI}$,
    - and augmenting of $CG \rightarrow H$ to infer $\mathbf{CGI \rightarrow HI}$,
    - and then transitivity: $CG \rightarrow CGI \rightarrow HI$

2. Closure of an attribute set

- Given a set of attributes α and a set of FDs *F*, ***closure of*** α ***under F*** is the set of all attributes implied by α
- In other words, the largest β such that: $\alpha \rightarrow \beta$
- Redefining *super keys:*
  - *The closure of a super key is the entire relation schema*
- Redefining *candidate keys:*
  1. *It is a super key*
  2. *No subset of it is a super key*

Computing the closure for α

- Simple algorithm:

1. Start with $\beta = \alpha$.
2. Go over all functional dependencies, $\delta \rightarrow \gamma$, in F^+
3. If $\delta \subseteq \beta$, *then*

   add γ to β

- 4. Repeat till β stops changing

Example

- *R = (A, B, C, G, H, I)*
  F = { $A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H$}

- $(AG)^+$?
  - 1. β = AG
  - 2. β = ABG (A → B and A ⊆ AG)
  - 3. β = ABCG (A → C and A ⊆ ABG)
  - 4. β = ABCGH (CG → H and CG ⊆ ABCG)
  - 5. β = ABCGHI (CG → I and CG ⊆ ABCGH)
  - *done*

- Is (AG) a candidate key ?
- It is a super key.
- (A+) = ABCH, (G+) = G.
- *YES.*

Uses of attribute set closures

- Determining ***superkeys and candidate keys***
- Determining if $\alpha \rightarrow \beta$ is a valid FD
  - Does $\alpha+$ contain β ?
- Can be used to compute F^+

3. Extraneous Attributes

Consider *F*, and a functional dependency, $\alpha \rightarrow \beta$.

- "Extraneous": Any attributes in α *or* β that can be safely removed ?

  without changing the constraints implied by F

- σ is *extraneous* in α iff:
  1. σ is in α, and
     - *F logically implies F'*
     - *where* $F' = (F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - \sigma) \rightarrow \beta\}$ *(i.e., show that F implies* $(\alpha - \sigma) \rightarrow \beta$*)*
  2. or show $(\alpha - \sigma)^+$ includes β under *F*
- σ is *extraneous* in β if:
  1. σ is in β, and
     - *F'* logically implies F,
     - $F' = (F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - \sigma)\}$
  2. or show α^+ includes σ under *F'*

σ is *extraneous* in α iff:
  F → F', *or*
  $(\alpha - \sigma)^+$ includes β under *F*

σ is *extraneous* in β iff:
  F' → F, *or*
  α^+ includes σ in *F'*

3. Extraneous Attributes

Must prove!

- Example: Given $F = \{A \to C, AB \to CD\}$, *show* **C extra in AB → CD**
  - F' = {A → C, **AB → D**} ← using

  - Using Armstrong's : (show F' → F)
    - We know:
      - AB → D (F')
      - ABC → CD (aug)
    - also:
      - A → C (F')
      - AB → BC (aug w/ B)
      - AB → ABC (aug w/ A)
    - then:
      - AB → ABC → CD (trans)

    done.

  - ***Or*** using attribute closures (show α+ includes C under F'):
    - We know:
      - AB → AB (reflexive)
      - AB → ABC (because A → C)

σ is *extraneous* in α iff:
F → F', *or*
$(\alpha - \sigma)^+$ includes β under *F*

σ is *extraneous* in β iff:
F' → F, *or*
α^+ includes σ in *F'*

3. Extraneous Attributes

- Example: Given *F* = {A → BE, B → C, C → D, AC → DE}, *using attribute closures*
  - For left side of AC → DE
    - A extraneous?
      - does C^+ include DE under F?
      - NO: C^+ = CD, NOT include DE
    - C extraneous?
      - does A^+ include DE under F?
      - YES: A^+ = ABCDE
  - Now F = A → BE, B → C, C → D, **A → DE**
    - B extraneous in A → BE?
      - F' = **A → E**, B → C, C → D, A → DE
      - Does A^+ include B under F'?
      - NO: A^+ = ADE
    - E extraneous in A → BE?
      - F' = **A → B**, B → C, C → D, A → DE
      - Does A^+ include E under F'?
      - YES: A+ = ABCDE
  - Now F = **A → B**, B → C, C → D, A → DE
    - D extraneous in right side of A → DE?
      - F' = A → B, B → C, C → D, **A → E**
      - Does A^+ include D under F'?
      - YES: A^+ = ABCDE
  - Now F = A → B, B → C, C → D, **A → E**

σ is *extraneous* in α iff:
F → F', *or*
$(\alpha - \sigma)^+$ includes β under *F*

σ is *extraneous* in β iff:
F' → F, *or*
α^+ includes σ in *F'*