

A weak entity can be identified uniquely only by

considering the primary key of some other (owner) entity.

- Owner entity set and weak entity set must participate in a one-to-n relationship set (one owner, many weak entities).

- Weak entity set must have *total* participation in this *identifying* relationship set.

- Dependent identifier is unique only within owner context

- Overlap constraints: Can employee Joe be an Hourly\_Emps as well as a Contract\_Emps entity? (disjoint if not)

- Covering constraints: Must every Employees entity be either an Hourly\_Emps or a Contract\_Emps entity? (covering if so)

Saying K is a candidate key for R means K -> R

Armstrong’s Axioms (X, Y, Z are sets of attributes):

§ Reflexivity: If X Y, then Y -> X

§ Augmentation: If X -> Y, then XZ -> YZ for any Z

§ Transitivity: If X -> Y and Y -> Z, then X -> Z

If X is part of a (candidate) key, we will say that

X is a prime attribute.

If X (an attribute set) contains a candidate key,

we will say that X is a superkey.

Rel’n R is in 1NF if all of its attributes are atomic.

§ No set-valued attributes! (1NF = “flat”)

§ Usually goes w/o saying for relational model

Rel’n R is in 2NF if it is in 1NF and no non-prime

attribute is partially dependent on a candidate key

of R.

Rel’n R is in 3NF if it is in 2NF and it has no

transitive dependencies to non-prime attributes.

Rel’n R with FDs F is in BCNF if, for all X -> A in F+

- A X (trivial FD), or else

- X is a superkey (i.e., X contains a key) for R.

- A is part of some key for R (i.e., it’s a prime attribute).

In other words, R is in BCNF if the only non-trivial

FDs that hold over R are key constraints (key -> attr)

There are three potential “problems” to consider:

1. Some queries will become more expensive.

2. Given instances of the decomposed relations, we may

not be able to reconstruct the corresponding instance

of the original relation! (If “lossy”...)

3. Checking some dependencies may require joining the

instances of the decomposed relations.

CREATE TABLE cs122a\_hw.Recordings(

recording\_id integer NOT NULL,

start\_time date,

end\_time timestamp,

meeting\_id integer NOT NULL,

PRIMARY KEY(recording\_id),

UNIQUE(start\_time)

FOREIGN KEY(meeting\_id) REFERENCES cs122a\_hw.Meetings ON DELETE CASCADE

);

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Text, letter

Description automatically generated

Text

Description automatically generated

CREATE PROCEDURE RegisterInstructor(

    IN user\_id text,

    IN email text,

)

LANGUAGE PLPGSQL AS $$

    BEGIN

        INSERT INTO swoosh.Users

        VALUES(user\_id, email, first\_name, last\_name);

        IF title IS NOT NULL THEN

            INSERT INTO swoosh.instructor

            VALUES(user\_id, title);

        END IF;

    END;

$$ ;

Graphical user interface, text

Description automatically generated

Text

Description automatically generatedText

Description automatically generated with medium confidence

ALTER TABLE swoosh.recording DROP CONSTRAINT recording\_meeting\_id\_fkey;

CREATE FUNCTION AddYoungSailors()

RETURNS Trigger

AS $$

BEGIN

IF ((SELECT MIN(age) FROM NewSailors) >= 10) THEN

INSERT INTO YoungSailors (sid, sname, age, rating)

SELECT sid, sname, age, rating FROM NewSailors

WHERE age < 18;

ELSIF ... THEN do\_something();

ELSE

RAISE 'Sailors must be at least 10';

END IF;

RETURN NEW;

END;

$$

LANGUAGE PLPGSQL;

CREATE TRIGGER YoungSailorLogger2

AFTER INSERT ON Sailors

REFERENCING NEW TABLE AS NewSailors

FOR EACH STATEMENT

EXECUTE FUNCTION AddYoungSailors();

LANGUAGE PLPGSQL;

CREATE TRIGGER YoungSailorLogger2

AFTER INSERT ON Sailors

REFERENCING NEW TABLE AS NewSailors

FOR EACH STATEMENT

EXECUTE FUNCTION AddYoungSailors();

Text

Description automatically generatedA picture containing text

Description automatically generatedA screenshot of a computer

Description automatically generated with low confidenceTable

Description automatically generatedText

Description automatically generated

Text

Description automatically generatedText

Description automatically generated

Text

Description automatically generated

Letter

Description automatically generated

Text

Description automatically generatedText

Description automatically generatedText, letter

Description automatically generatedText

Description automatically generatedText, letter

Description automatically generatedText, letter

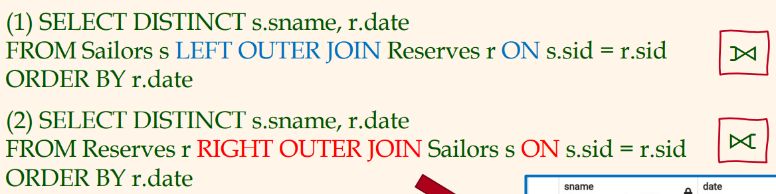
Description automatically generatedText

Description automatically generated

Text

Description automatically generatedText

Description automatically generated

Text

Description automatically generated with low confidenceText

Description automatically generatedGraphical user interface, text, application

Description automatically generatedText, letter

Description automatically generated