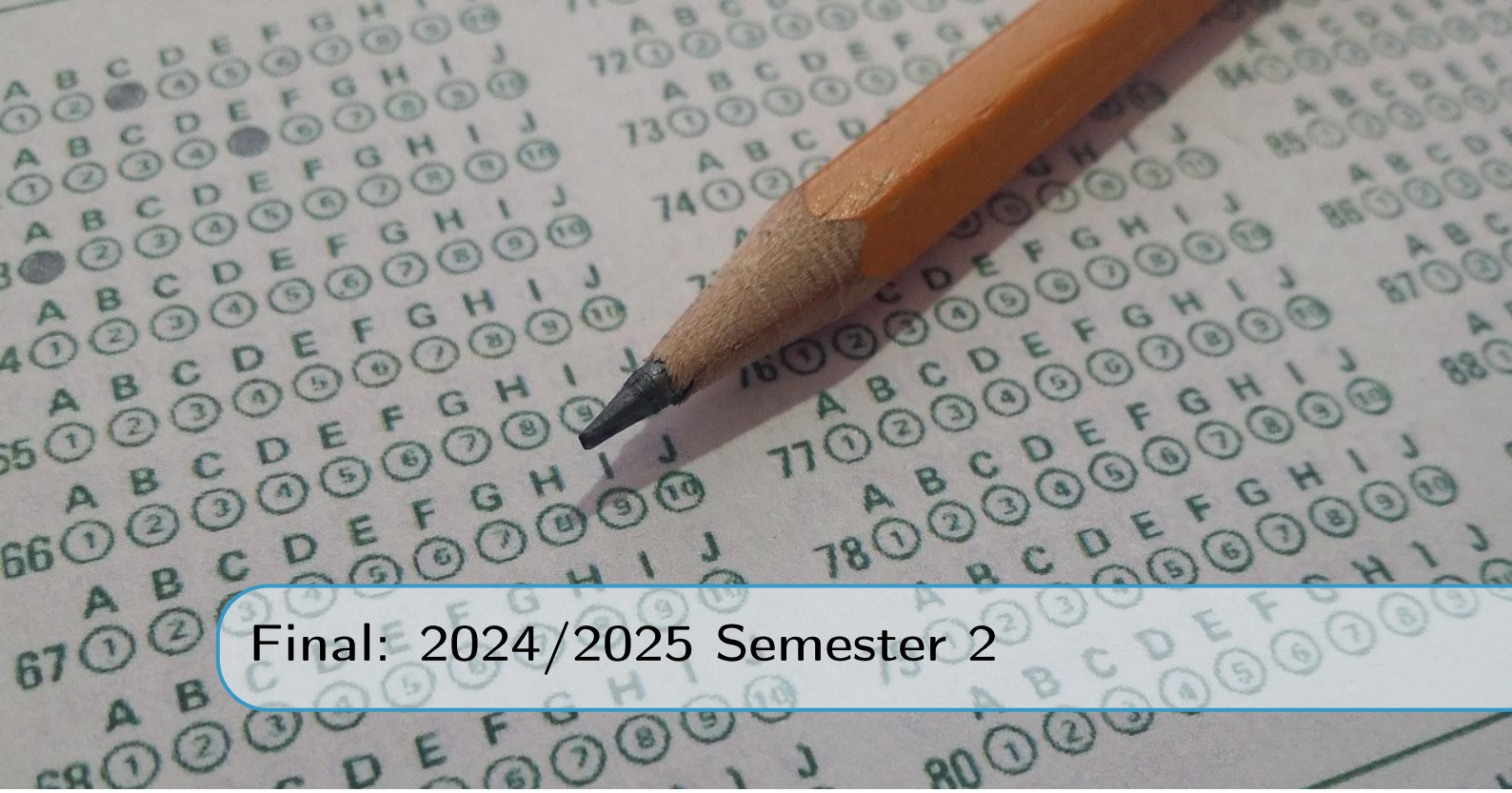
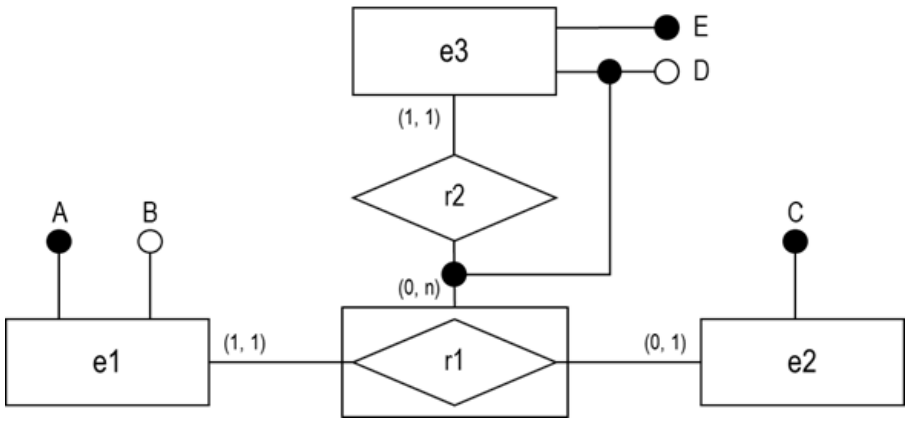


Final: 2024/2025 Semester 2

For the first 5 questions, we will be using the following entity-relationship diagram.



1. Entity-Relationship Diagram and Functional Dependencies.

   Notes: E

   $\{C\} \to \{B\}$ $\{E\} \to \{A\}$ $\{E\} \to \{B\}$ none holds.

2. Entity-Relationship Diagram and Keys.

   Notes: ABD

   The keys are $\{E\}$ $\{A, D\}$ $\{C, D\}$.

3. Entity-Relationship Diagram and BCNF.

   Notes: BD

   $\{A, B, C\}$ $\{C, D, E\}$. Note that $\{B, C, D\}$ is not in BCNF.

4. Entity-Relationship Diagram and 3NF.

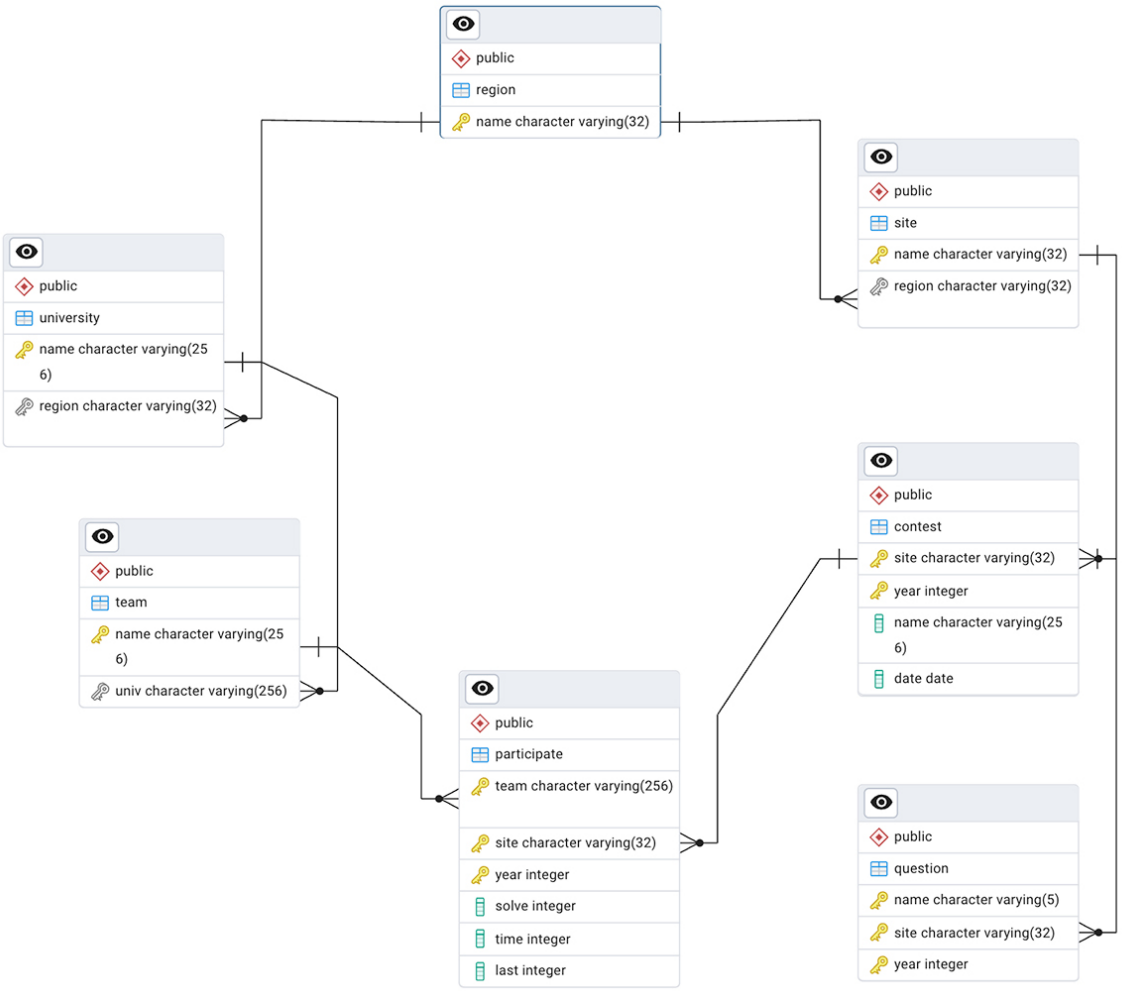
   Notes: BD

   $\{A, B, C\}$ $\{C, D, E\}$. Note that this schema prevents us from querying any entity `e2` that are not participating in `r1`. To get that, we look at the schema translation in the next question.

5. Entity-Relationship Diagram and Schema Translation.

> **Notes: ABD**
>
> $\{C\}$ $\{A, B, C\}$ $\{C, D, E\}$. Notice how schema translation match closely with BCNF and 3NF decomposition. The addition of $\{C\}$ is because there is no *non-trivial* functional dependencies in `e2`.

For the next 3 questions, we will be using the following logical diagram.



6. Relational Algebra.

   Find all the different team names that have participated in two different sites (only site, not including year).

> **Notes: ACD**
>
> This is about equivalence of join condition with selection operation. Note that if there is no condition then it is a *natural join* which does not work for our logical diagram.

7. Relational Algebra.

   Find all the different team name that solves the most number of problems in the year 2023.

> **Notes: B**
>
> This question highlights the difficulties of dealing with outer join. The idea of rows that **do not match** anything on the other table is critical as it defines what will be paired with the value `NULL`.
>
> The condition on option **B** pair with `NULL` all the values from `p1` that is NOT smaller than any value from `p2`. This is exactly what we need.

8. Relational Algebra.

   Find all the different university names that has no team participating in any contest in the year 2024.

> **Notes: CD**
>
> This is similar to the previous question but we use `EXCEPT` instead.

For the next 5 questions, the questions are on stored procedures, triggers, and Python programming.

9. Top N Students with 1-2-2-3 Ranking.

> **Notes:**
>
> The following is just one possible answer. There are other alternative answers related to the combination for blank 4 and 5. We check with running against an extensive test cases.

```
CREATE OR REPLACE FUNCTION top_n(n INT)
RETURNS SETOF scores AS $$ /* 1 */
DECLARE
  curs CURSOR FOR (SELECT * FROM scores ORDER BY mark DESC);
  r RECORD;  prev INT;  current_rank INT;
BEGIN
  prev := -1;
  current_rank := 0; /* 2 */
  OPEN curs;

  LOOP
    FETCH curs INTO r;
    EXIT WHEN NOT FOUND; /* 3 */

    IF prev <> r.mark THEN /* 4 */
      current_rank := current_rank + 1;
    END IF;

    prev := r.mark;
    EXIT WHEN current_rank > n; /* 5 */

    RETURN NEXT r; /* 6 */
  END LOOP;
  CLOSE curs;
END;
$$ LANGUAGE plpgsql;
```

10. Car Park Management System - Constraint Enforcement.

> **Notes:**
>
> The following is just one possible answer. We check with running against an extensive test cases.

```
CREATE OR REPLACE FUNCTION sp_check1()
RETURNS TRIGGER AS $$
BEGIN
  IF EXISTS (SELECT 1 FROM seasons WHERE car_plate = NEW.CAR_PLATE) THEN /* 1 */
    RAISE NOTICE 'Car plate exists in the season parking table!';
    RETURN NULL; /* 2 */
  END IF;

  IF EXISTS (SELECT 1 FROM visitors
              WHERE car_plate IN (NEW.car_plate, OLD.CAR_PLATE) /* 3 */
                AND time_out IS NULL AND NEW.time_out IS NULL) THEN
    RAISE NOTICE 'Immutable non-exited row!';
    RETURN NULL; /* 4 */
  END IF;

  RETURN NEW; /* 5 */
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER tk1_on_visitors
BEFORE INSERT OR UPDATE ON visitors  /* 6 */  /* 7 */
FOR EACH ROW EXECUTE FUNCTION sp_check1();
```

```
ALTER TABLE visitors
ADD CONSTRAINT chk_sh_plate_format
CHECK (
  substr(car_plate, 1, 1) ~ '[A-Z]' AND (substr(car_plate, 1, 1) <> 'S' OR is_valid(
    car_plate)) /* 8 */
);
```

11. Python.

Notes: E

```
cursor.execute("SELECT * FROM scores WHERE mark = %s", (int(input()),))
```

12. Stored Procedure.

Notes: B

After `CALL f1(4, sum);` the value of the variable sum is 10.

13. Statement-Level Interface.

Notes: E

"None of the other options is correct."

For the next 3 questions, we will use the following relation R and set of functional dependencies S.

- $R(A, B, C, D, E, F, G)$
- $S = \{\{AB\} \to \{C\}, \{CD\} \to \{E\}, \{FG\} \to \{ABC\}, \{B\} \to \{AF\}, \{A\} \to \{BE\}\}$

14. Keys.

Notes:

$\{A, D, G\}$ $\{B, D, G\}$ $\{D, F, G\}$

15. BCNF Decomposition.

Notes:

One possible decomposition is: $R_1(ABCEF)R_2(ADG)$.

Grading is done by checking the following properties.

- No missing attributes (*i.e.,* valid decomposition).
- All decomposed relations in BCNF.
- The decomposition is a lossless-join decomposition.

16. Dependency Preserving Decomposition.

Notes:

Using $R_1(ABCEF)R_2(ADG)$: **FGHI**. The functional dependencies are: $\{CD\} \to \{E\}$, $\{FG\} \to \{A\}$, $\{FG\} \to \{B\}$, and $\{FG\} \to \{C\}$.

Grading is done by checking against the actual decomposition from previous question. However, we require the previous question to be correct. Otherwise, it is easy to simply put blanks for the previous question and answer that all functional dependencies are not preserved.

For the next 3 questions, we will use the following relation R and set of functional dependencies S.

- $R(A, B, C, D, E, F)$
- $S = \{\{D\} \to \{B\}, \{AB\} \to \{C\}, \{BC\} \to \{A\}, \{CD\} \to \{E\}, \{BD\} \to \{E\}, \{ADE\} \to \{C\}\}$

17. Keys.

Notes:

$\{A, D, F\}$ $\{C, D, F\}$

18. Minimal Basis.

Notes:

One possible minimal basis is: $\{\{AB\} \to \{C\}, \{BC\} \to \{A\}, \{D\} \to \{B\}, \{D\} \to \{E\}\}$.

Grading is done by checking the following properties.

- The answer is *equivalent* to S.
- The answes is a minimal basis of itself (*note that the check is for each property for completeness*).

19. 3NF Decomposition.

Notes:

Using $\{\{AB\} \to \{C\}, \{BC\} \to \{A\}, \{D\} \to \{B\}, \{D\} \to \{E\}\}$, the decomposed relations are: $R_1(A, B, C), R_2(B, D, E)$. We also add one of the following from the key: $R_3(ADF)$ or $R_3(CDF)$.

Grading is done by checking against the actual minimal basis from previous question. However, we require the minimal basis to at least be equivalent. We then check the following properties related to 3NF decomposition.

- No missing attributes (*i.e.,* valid decomposition).
- All decomposed relations in BCNF.
- The decomposition is a lossless-join decomposition.
- The decomposition is a dependency preserving decomposition.

The last question is a miscellaneous question.

20. 3NF vs BCNF.

Notes: It is possible to construct an answer involving 6 functional dependencies.

$$\{\{AB\} \to \{C\}, \{AC\} \to \{D\}, \{AD\} \to \{B\}, \{BC\} \to \{D\}, \{BD\} \to \{C\}, \{CD\} \to \{B\}\}$$

Grading is done by checking all the properties.

- R is not in BCNF.
- R is in 3NF.
- S is a minimal basis of itself.

Heavy penalty is imposed for each violation. Since there are 5 marks for the question, we expect at least 5 functional dependencies with a penalty of -1 mark for each missing functional dependencies.