CS698F Advanced Data Management

Instructor: Medha Atre

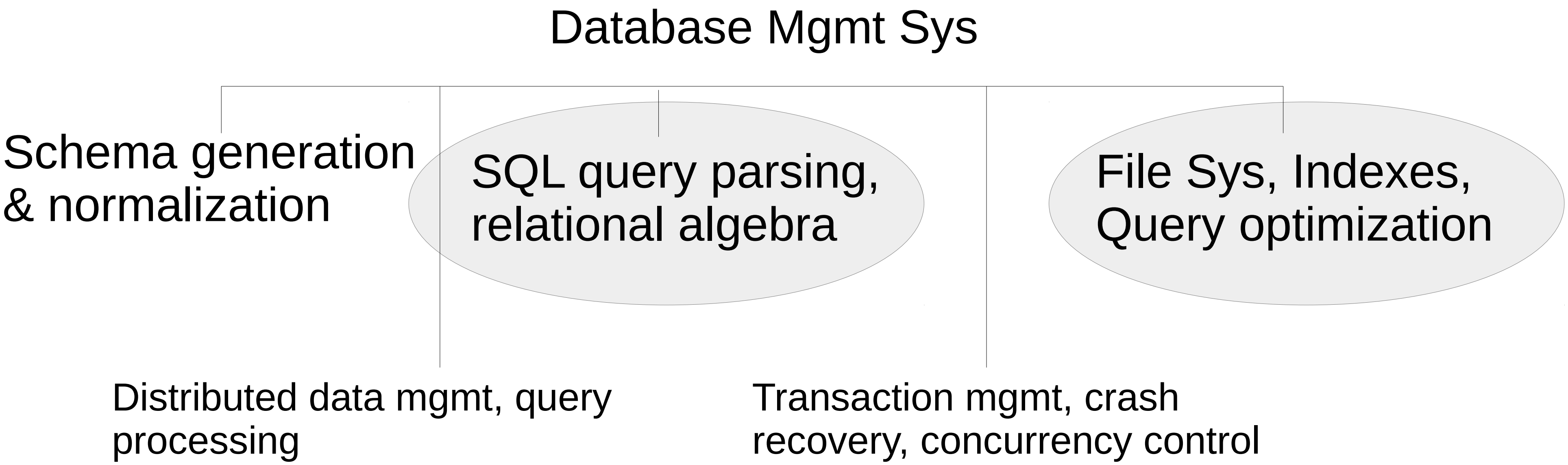
Grading Scheme

- Reading Assignment-1: 10%
  - Pre-midterm, first week of September
  - Choose a paper from list of papers floated by instructors, read that paper and its relevant other papers and present in the class (with proper PPT/PDF presentation)
- Mid-semester: 20%
  - Presentation of literature survey for course project and course project intermediate demo
- Reading Assignment-2: 10%
  - Pre-endsem, last week of October
  - Choose a paper from list of papers floated by instructors, read that paper and its relevant other papers and present in the class (with proper PPT/PDF presentation)

Grading Scheme

- Course project impl and demo: 30%
  - Last week of classes (before endsem)
- Course project written report: 10%
  - Due last week of classes (before endsem)
- Endsem exam (written): 20%
  - Questions asked on the papers read throughout the semester and topics covered in the classes.
    - Hence understand the papers you read and present well.

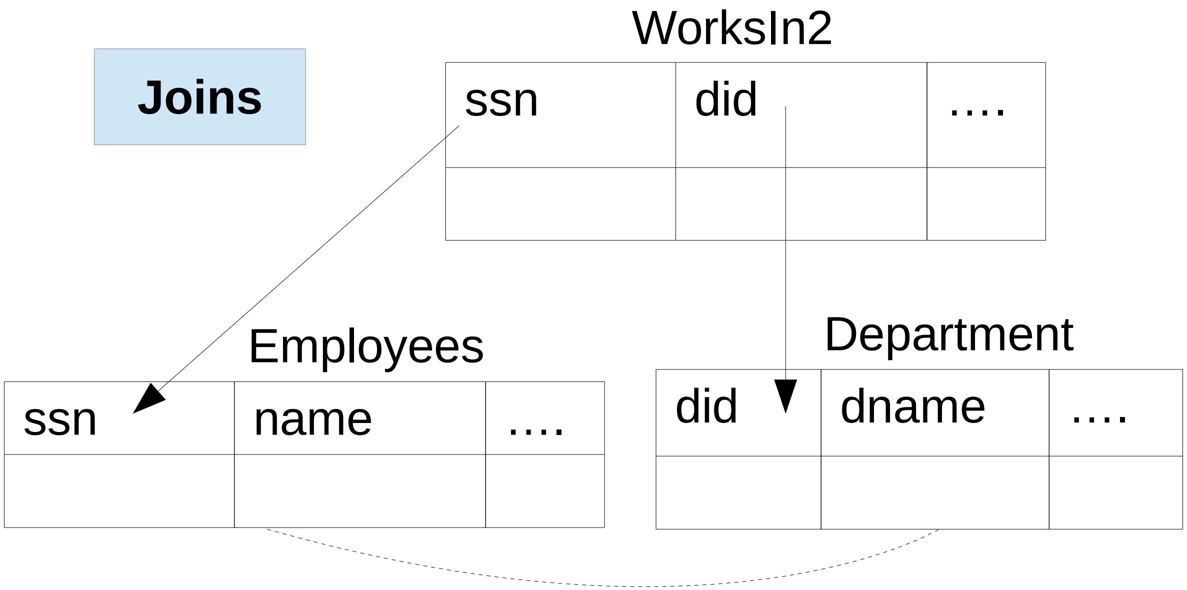
Recap


Database Mgmt Sys
Schema generation & normalization
SQL query parsing, relational algebra
File Sys, Indexes, Query optimization
Distributed data mgmt, query processing
Transaction mgmt, crash recovery, concurrency control

SQL queries

SELECT E.name, D.dname
FROM WorksIn2 as W,
Employees as E,
Department as D
WHERE
W.ssn=E.ssn AND
W.did="CSE" AND
W.did=D.did

Joins



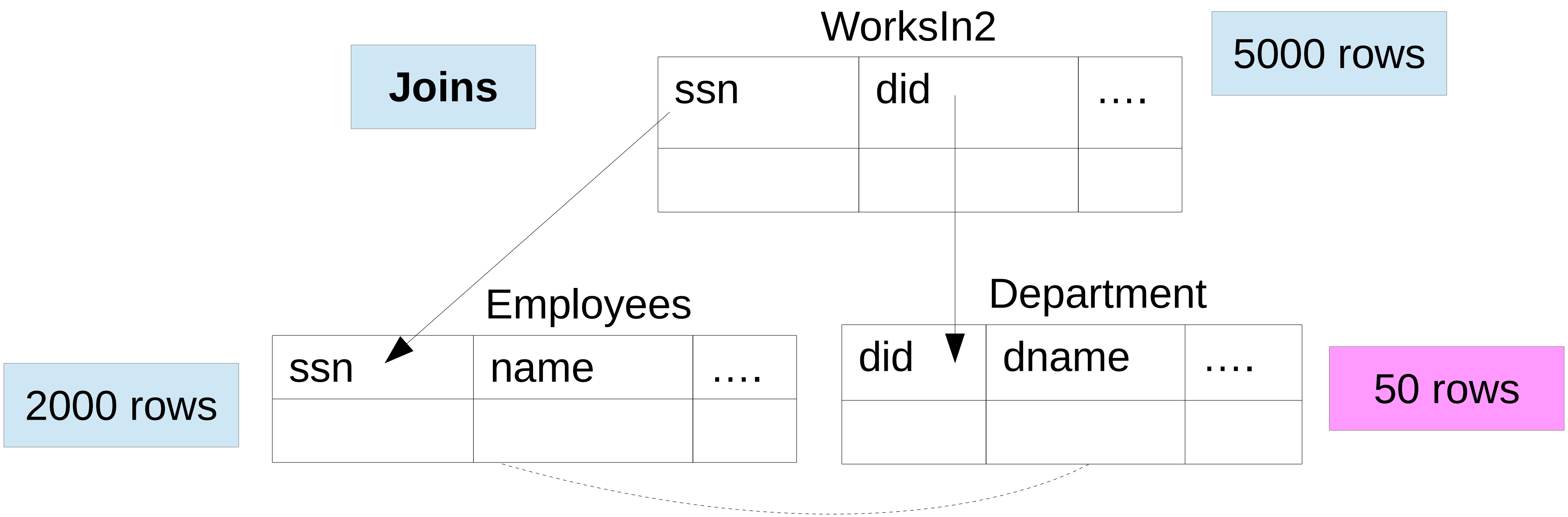
Relational Algebra

- *Algebraic* representation of the SQL queries.
  - $\Pi_{(E.name,\ D.dname)}$ (W ⋈ E ⋈ D)
  - Π Projection symbol (what you are *SELECTing*)
  - ⋈ Join symbol (what tables are in *FROM* clause)
  - Conditions are in the *WHERE* clause
    - Selection conditions: $\sigma_{(D.did\ =\ "CSE")}$
    - Join conditions: $\bowtie_{(W.ssn\ =\ E.ssn, W.did\ =\ D.did)}$

Natural Joins 101

- Allows algebraic and set-theoretic operations
  - Helps in finding variety of *query plans* (we will visit this)
  - $(W \bowtie E \bowtie D) \equiv (E \bowtie W \bowtie D) \equiv ((W \bowtie E) \bowtie D)$
  - Joins are commutative and associative
- SQL has other set like operations as UNION
  - $(W1 \cup W2) \bowtie (E1 \cup E2) \equiv (W1 \bowtie E1) \cup (W1 \bowtie E2) \cup (W2 \bowtie E1) \cup (W2 \bowtie E2)$

Cost estimation



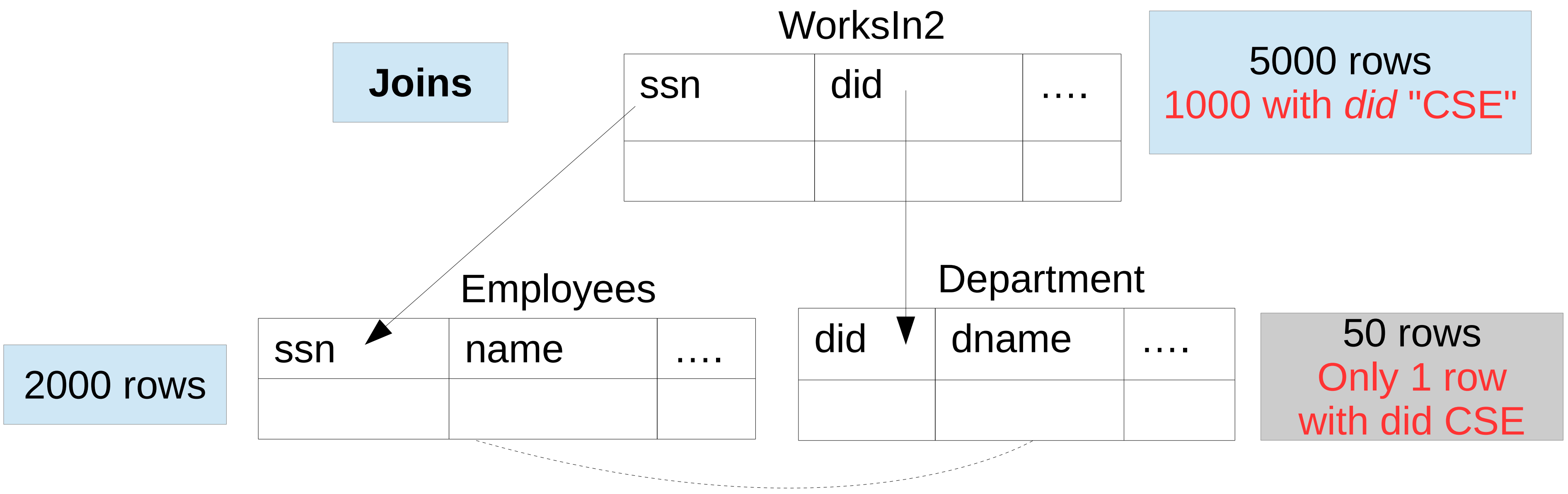
Cost estimation 101

- Joins are commutative and associative
  - Alternatives for performing (W ⋈ E ⋈ D)
  - First R1= (W ⋈ E) (5000 x 2000) and then R2 = (R1 ⋈ D) (5000 x 2000 x 50) in reality this number will be much lower due to *join selectivity*.
  - First R1= (W ⋈ D) (5000 x 50) and then R2 = (R1 ⋈ E)
  - First R1 = (E ⋈ D) (2000 x 50) and then R2 = (R1 ⋈ W)
- So would you choose (E ⋈ D)?
  - No! Because E and D do not share any attribute
  - It is a Cartesian product, no reduction in result size due to *join selectivity*.

Cost estimation 101

- Join selectivity is *how many results* will actually get generated?
  - Never equal to Cartesian product! Why?
- Usually not every column value appears on both the sides.
- Even if it does, simple math proves that join results will never be equal to the Cartesian product!
  - $(W \bowtie D) \neq W \times D$ (x => Cartesian product)

How selections affect



Query Optimization 101

- Query rewriting a.k.a. considering various query plans for the *same effective results*.
  - Relational algebraic equivalences help
- Indexes on the tables a.k.a. access methods
  - Types of indexes – B+ trees, Hash index, others we will see in the contexts of different data types.
- Join methods and their costs
  - Nested-loop, sort-merge, index-nested-loop join, hash join etc.
- Finally combining the above together for cost optimization.

Query Optimization 101

- Query rewriting a.k.a. considering various query plans for the *same effective results*.
  - Relational algebraic equivalences help
- Indexes on the tables a.k.a. access methods
  - Types of indexes – B+ trees, Hash index, others we will see in the contexts of different data types.
- Join methods and their costs
  - Nested-loop, sort-merge, index-nested-loop join, hash join etc.
- Finally combining the above two together for cost optimization.

Relational algebra rules

- Natural joins are commutative and associative
  - $(W \bowtie E \bowtie D) \equiv (E \bowtie W \bowtie D) \equiv ((W \bowtie E) \bowtie D)$ (you are advised to avoid Cartesian products)
- Join conditions are equivalent to selections over Cartesian product, but not over joins
  - $(W \bowtie_{w.ssn=e.ssn} E) \equiv \sigma_{w.ssn=e.ssn}(W \times E) \neq \sigma_{w.ssn=e.ssn}(W \bowtie E)$
- Selections are idempotent and commutative
  - $\sigma_{w.did="CSE"}(W) \equiv \sigma_{w.did="CSE"}(\sigma_{w.did="CSE"}(W))$
  - $\sigma_{w.did="CSE"}(\sigma_{w.ssn="1234"}(W)) \equiv \sigma_{w.ssn="1234"}(\sigma_{w.did="CSE"}(W))$

Relational algebra rules

- Conjunctive and Disjunctive selections (recall set theory, and boolean ops)
  - $\sigma_{w.did="CSE" \land w.ssn="1234"} (W) \equiv \sigma_{w.did="CSE"} (\sigma_{w.ssn="1234"}(W))$
  - $\sigma_{w.did="CSE" \lor w.ssn="1234"} (W) \equiv \sigma_{w.did="CSE"}(W) \cup \sigma_{w.ssn="1234"}(W)$
  - $\sigma_{w.did="CSE"} (W1 \cup W2) \equiv \sigma_{w.did="CSE"}(W1) \cup \sigma_{w.did="CSE"}(W2)$
  - $\sigma_{w.did="CSE"} (W1 \cap W2) \equiv \sigma_{w.did="CSE"}(W1) \cap \sigma_{w.did="CSE"}(W2)$

Relational algebra rules

- Selections and Projects combined
  - $\Pi_{\text{w.did, w.ssn}} (\sigma_{\text{w.did="CSE"}}(W)) \equiv \sigma_{\text{w.did="CSE"}} (\Pi_{\text{w.did, w.ssn}} (W))$
  - That is to say, if the *projected* columns/attributes are superset of selection attributes, then projection and selection can be interchanged!
    - $\{\text{w.did, w.ssn}\} \supseteq \{\text{w.did}\}$
- Many more rules too, but for our purpose these suffice.

Back to our query

SELECT E.name, D.dname
FROM WorksIn2 as W, Employees as E, Department as D
WHERE
    W.ssn=E.ssn AND W.did="CSE" AND W.did=D.did

$$\Pi_{E.name,\ D.dname}\ ((\sigma_{did="CSE"}(W \bowtie_{ssn} E)) \bowtie_{did} D)$$

Push selection inside

SELECT E.name, D.dname
FROM WorksIn2 as W, Employees as E, Department as D
WHERE
W.did="CSE" AND W.ssn=E.ssn AND W.did=D.did

$$\Pi_{E.name,\ D.dname}((\sigma_{did="CSE"}(W)) \bowtie_{ssn} E) \bowtie_{did} D)$$

Observe again

SELECT E.name, D.dname
FROM WorksIn2 as W, Employees as E, Department as D
WHERE
W.did="CSE" AND W.ssn=E.ssn AND W.did=D.did

$$\Pi_{E.name,\ D.dname}((\sigma_{did="CSE"}(W)) \bowtie_{ssn} E) \bowtie_{did} D)$$

Apply commutative property

SELECT E.name, D.dname
FROM WorksIn2 as W, Employees as E, Department as D
WHERE
W.did="CSE" AND W.did=D.did AND W.ssn=E.ssn

$$\Pi_{E.name,\ D.dname}\left(\left(\sigma_{did="CSE"}(W)\right) \bowtie_{did} D\right) \bowtie_{ssn} E)$$

Observe again

SELECT E.name, D.dname
FROM WorksIn2 as W, Employees as E, Department as D
WHERE
W.did="CSE" AND W.did=D.did AND W.ssn=E.ssn

$$\Pi_{\text{E.name, D.dname}} ((\sigma_{\text{did="CSE"}}(W)) \bowtie_{\text{did}} D) \bowtie_{\text{ssn}} E)$$

Can we push projection inside? Let us see?

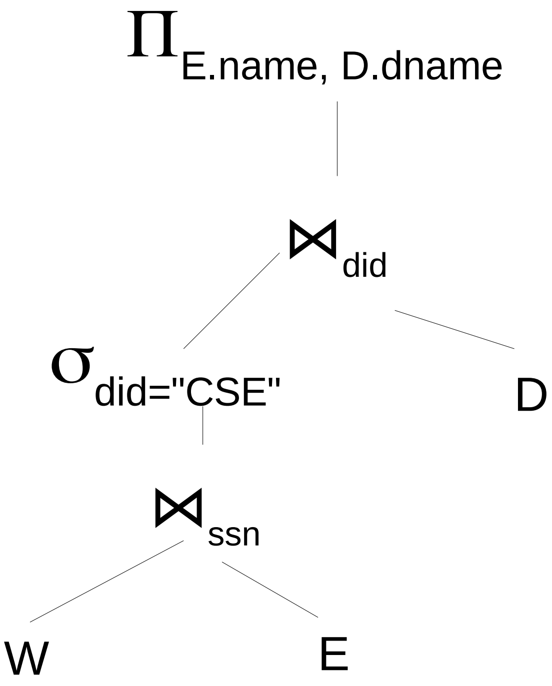
Cannot push projection inside

SELECT E.name, D.dname
FROM WorksIn2 as W, Employees as E, Department as D
WHERE
W.did="CSE" AND W.did=D.did AND W.ssn=E.ssn

$$\sigma_{did="CSE"} (\Pi_{E.name,\ D.dname} ((W \bowtie_{did} D) \bowtie_{ssn} E))$$

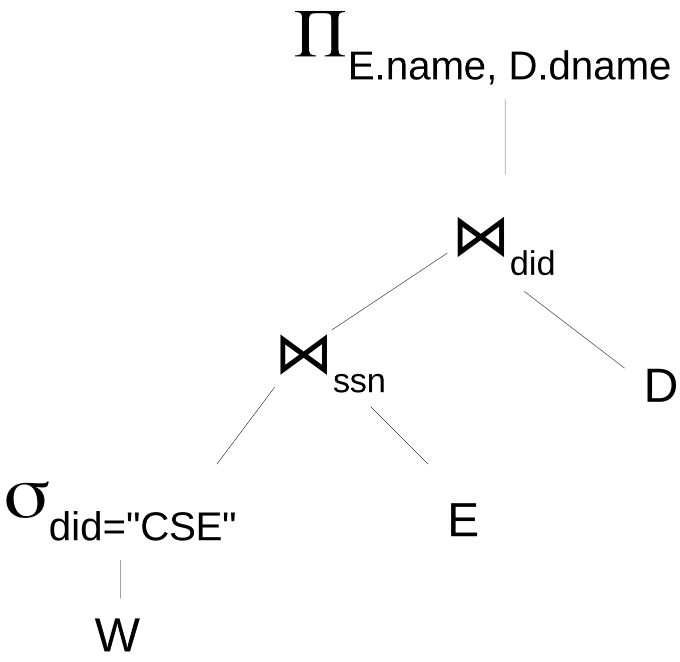
Query operator tree-1

SELECT E.name,
D.dname
FROM WorksIn2 as W,
Employees as E,
Department as D
WHERE
W.did="CSE" AND
W.did=D.did AND
W.ssn=E.ssn



Query operator tree-2

SELECT E.name, D.dname
FROM WorksIn2 as W, Employees as E, Department as D
WHERE
W.did="CSE" AND W.did=D.did AND W.ssn=E.ssn



Query operator tree-3...4… so on

SELECT E.name, D.dname
FROM WorksIn2 as W, Employees as E, Department as D
WHERE
    W.did="CSE" AND W.did=D.did AND W.ssn=E.ssn

Draw at home as an exercise!

Next...

- Overview of indexes.
- Incorporating them into query plans.
- Floating of papers for assignments and course project.
- Deadline for course project selection: 19 August, 2017 11:59pm.