DBMS

CLASSTIME | Page No.
Date / /

Week-1

L1.1 Course Overview

- DBMS contains info. abt. a particular enterprise
  - collecⁿ of interrelated data
  - set of programs to access the data
  - convenient and efficient envt.
- DB applicatⁿ — everywhere
- they can be very large, and touch all aspects of our lives
- University DB → time tables, GPAs
- these applicaⁿs were built on file system
- drawbacks of using file sys. to store data
  (i) data redundancy and inconsistency
  (ii) difficulty in accessing data
  (iii) data isolaⁿ
  (iv) integrity problems - integrity constraints become "buried" in program code, hard to add new constraints
  (v) atomicity of updates
  (vi) concurrent access by multiple users
  (vii) security problems

  ↓

  DB sys. offer solⁿs to all these problems

- Pre req.
  → set thy., de morgan's laws, relaⁿ & funcⁿ
  → NPTEL → MOOCs : Discrete maths
  → propositional logic / boolean algebra
  → predicate logic (quantifiers), data struc. (array, list, B- tree, hash tables, etc.)
  → algos and prog. in C → NPTEL (intro to C)
  → OO analysis and design
- DB sys. concepts → 6ᵗʰ edition → Silberschatz

CLASSTIME | Page No.
Date / /

L1.2 Why DBMS ? 1

- to understand DBMS from historical perspective
- mgmt. of data / records is a basic need of human society — storage, retrieval, transac", audit, archival
  For - indi, small / big enterprise, global
  Approaches - Physical, electronic
- book keeping - using physical ledgers and journals for centuries
- Henry Brown - Nov 2, 1886 - Receptacle for storing and preserving papers
- Herman Hollerith, 1890 - punch cards for memory
- 1950s - CP started ; 1960s - data mgmt. with punch cards ; 1970s - COBOL, CODASYL, VisiCalc ; 1980s - RDBMS ; 2000s - NoSQL ; 2010s - DS↑
- DData mgmt depends on parameters - Speeds, Scale, security, retrieval, ease of use, consistency, efficiency, cost ...
- book keeping problems
- spreadsheet files - useful for single user or small enterprise applica"
- the above limita" of filesystems paved the way for a comprehensive platform dedicated to data — DBMS
- 1960s & 1970s → Ted Codd - Relational data model → SQL, 1980s → 2000s, XML & XQuery → later, Google BigTable, Yahoo PNuts, Amazon

L1.3 Why DBMS ? 2

- case study on a bank transac" -

'a+' ⇒ append mode

CLASSTIME | Page No.
Date / /

- file based (.csv) sys. using Python
- Python DBMS
→ very difficult to handle strings in-built features
→ diff. to Δ the struc. use simple SQL queries
→ in s in ms
→ implementa^n difficulty user specific access at DB level
→ arithmatic op^ns are easy not easy
→ low costs high costs

- read parameterized comparison yourself

L1.4 Intro to DBMS - 1

- phy. level - describes how a record is stored
- logical level. - describes data stored in DB, and relationships among data
- view level. - app. prog. hide details of data types
- schema → type of var } analogy
  instance → value of var. }
- logical schema - overall logical struc. of DB
  Eg. DB consists of info. about a set of customers
- Phy. schema - overall phy. struc. of DB
- Instance - actual content of the DB at a particular point in time.
- Phy. data indep. - the ability to modify the phy. schema w/o Δing the logical schema. Indep. of interface & implementa^n in OOP.
- data models - a collec^n of tools for describing data, data relationships schematics, constraints

- Relational model – all the data is stored in various tables
- DDL → data defini" lang. – specific nota" for defining DB schema . Eg. create
  ↳ The DDL compiler generates a set of table templates stored in data dictionary
- DML → Manipula" / Query lang.
  ↳ pure → relational algebra, calculus
  ↳ commercial → SQL
- SQL → widely commercial lang.
  → is not a Turing Machine equivalent lang
  → SQL or DB are used with other lang.
- Logical design – deciding on DB schema. DB design req. that we find a 'good' collec" of rela" schema.
- Phy. design – deciding on the phy. layout of the DB.

L1.5 Introduction to DBMS / 2

- need to come up with a methodology to ensure that each rela" in the DB is good
- 2 ways –
  ① entity relationship model (ER diag.)
  ② Normaliza" theory
- relational model: flat, atomic values
- Obj. relational data models –
  ① it includes Obj. orienta" and constructs to deal with added data types
  ② allow attributes of tuples to have complex types

③ preserve relational foundations, while extending modeling power
④ provide upward compatibility with existing relational languages

- XML : Extensible markup lang → W3C
  → originally document markup lang.
  → specify new tags & create nested tag struc
  → all new gen. data interchange formats
- DB Engine – storage manager, query processing and transac^n manager.
- Storage Mgmt. – interac^n with OS file manager efficient storing, retrieving and updating data
  Issues: ① storage access
  ② file organisa^n
  ③ indexing and hashing
- Query Processing – parsing and transla^n
  – Optimiza^n
  – evalua^n
- Transac^n mgmt. – collec^n of op^{ns} that performs a single logical func^n in DB applica^n. It ensures that the DB remains in a consistent state despite sys. failures and transac^n failures.
- The architecture of DBS is greatly influenced by underlying compt. sys.
  – centralized, client server, parallel, distributed, cloud

CLASSTIME | Page No.
Date / /

Week-2

L2.1 Intro to Relational Model 1

- Eg. of a rela^n → attributes (columns)
  → tuples (rows)
- The set of allowed values for each attribute is c/d a domain of the attribute
  Roll# → alphanum-string
  Dept./Name → alpha string
  DoB → date
  passport # → string
- attribute values are atomic (indivisible)
- the spl. value null is a member for every domain. Indicates that value is unknown
- The null value cause complica^n in the def^n of many languages.
- $A_1, A_2 \ldots A_n$ are attributes
  $R = (A_1, A_2 \ldots A_n)$ is a relation schema
  Eg. instructor = (ID, salary, ...)

  $D_1 \ldots D_n$ & relation r is subset of
  $D_1 \times D_2 \times \ldots \times D_n$
- The current values (relation instance) of a rela^n are specified by a table.
- An element t of r is a tuple, represented by a row in table.
- Order of tuples / rows is irrelevant. No 2 row/tuple may be identical.
- Let $K \subseteq R$ where R is the set of attributes in the rela^n. K is a superkey of R if values for K are sufficient to identify a unique tuple of each possible rela^n r(R)
- Superkey K is a candidate key if K is minimal

CLASSTIME | Page No.
Date / /

- One of the candidate key should be 1° key. A surrogate / synthetic key in a db is a unique identifier for either an entity in the modelled world or an obj. in the db.
  → does not come from applica" data unlike natural / business key coming from applica" data.

Eg. Aadhar or Roll No. for 1° key. Aadhar gives unique info. But, roll no. has additional useful info.

- 2° / alternate keys, simple key and comp. key
- Foreign key constraint: Value in 1 rela" must appear in another
- A comp. key consists of more than 1 attribute to uniquely identify an entity occurrence.
- Procedural prog. — eg. that prog. tells the comp. what to do. (how)
- Declarative prog. — more detailed style (what)
- pure language —— relational algebra

L 2.2 Intro. to Relational Model / 2

- Basic prop. of relations —
  ① rela" is a set. Ordering is not necessary, all rows / tuples must be distinct.
- select op" → selec" of rows.

$\sigma_{A=B \wedge D>5}(r)$ meaning (A = B and D > 5)

∧ ⇒ and
∨ ⇒ or

- project op" → selec" of columns (attributes)

$\Pi_{A,C}(r)$ → show A & C column

- union of 2 rela"

$r \cup s$ ⇒ join 2 tables

- set diff. of 2 relations
  $r - s$ → take out s things which are in r
- intersec^n of 2 rela^{ns}
  $r \cap s$ ⇒ common of r & s
  $r \cap s = r - (r - s)$
- joining 2 rela^{ns} — cartesian product
  $r \times s$ Eg. $2 \times 4 = 8$
- cartesian prod. ⇒ naming issues $r.B$ $s.B$
- $\rho_x(E)$ ⇒ returns the exp. E under the name X
  $r \times \rho_s(r)$
- composition - can build exp. using multiple opera^{ns}
- Natural join of R & S on RUS —
  (i) consider each pair of tuples t_r & t_s
  (ii) if t_r & t_s have same value on each attribute of RUS
- $r \bowtie s$

$$\Pi_{A, r.B, C, r.D, E}\left(\sigma_{r.b = s.B \wedge r.D = s.D}(r \times s)\right)$$

- aggregate operators — SUM, AVG, MAX, MIN
- relational algebra is turing comp.

L2.3 Intro to SQL | 1

- IBM — SEQUEL → SQL
- ANSI & ISO std. SQL
- DDL — allows specifica^n
- char, varchar, int, small int, numeric, float

- create table instructor (
  ID char(5),
  name varchar(20),
  dept_name varchar(20))
- integrity constraints – not null, 1° key, foreign key
- 1° key declaration on an attribute automatic ensures not null
- update tables
  insert (DML)
  delete (DML) ⇒ remove all tuples
  drop (DDL) ⇒ drop tables
  alter (DDL) ⇒ edit schema
- SQL query ⇒ select $A_1, A_2 \ldots$
  from $r_1, r_2 \ldots$
  where P
  result → relation
- select → list the attributes desired in the result of a query
- SQL allows duplicates in relⁿ in query results, if wants remove duplicates
  select distinct A ...
- where ⇒ specifies conditions
- from ⇒ scope of relations

L2.4 Intro to SQL / 2

- select * from instructor, teaches
  ↳ ID ← → instructor.ID
- not very useful, meaningless tuples
- select * from ins, teaches where
  ins.Id = teaches.ID

⇒ equi-join, Natural join

- select distinct T.name
  from instructor as T, instructor as S
  ↳ keyword as is optional
  ↳ instructor T
- '%' matches any substring
  '_' matches any character
  like '%dan' escape '\'
  ↳ we use '\' as the escape character.
- list in alphabetic order → order by
  'desc' or 'asc'
- 'select top' → specify the no. of records or limit to return
  or first first n rows only
- between → comparison operator
- in → allows you to specify multiple values in a where clause

L2.5 Intro to SQL | 3

- or ⇒ union
  in this and in this ⇒ intersect
  in this but not in this ⇒ except
- select distinct salary
  from instructor
  except
  select distinct T.salary
  from instructor T, S
  where T.salary < S.salary
  ↳ largest salary of all instructor
- to retain duplicates in multiset ⇒ union all, intersect all, except all.
- null ⇒ signifies an unknown value or that a value doesn't exist

- OR unknown or true = true
  unknown or false = unknown
  unknown or unknown = unknown

  AND true and unknown = unknown
  false and unknown = false
  unknown and unknown = unknown

  NOT not unknown = unknown
- aggregates → avg, max, min, add, count
- group by ⇒ several names, then it group by certain attribute
- for null values → count returns 0 and all other aggregates return null

Week-3

L3.1 SQL Examples

- select distinct c.building /all c.building
  from classroom c
  where c < 100;
- select name, budget
  from student s, dept d
  where s.dept_name = d.dept_name
  and budget < 100000
- select title
  from courses
  where course_id like '_ _ _ -%';
- select ——
  from ——
  where —— in ('comp', 'Stats');
- select building, avg(capacity)
  from classroom
  group by building
  having avg(capacity) > 25;
- select dept, sum(credits)
  from course
  group by dept.;

L3.2 Intermediate SQL / 1

- select $A_1, \ldots A_n$
  from $r_1, \ldots r_n$
  where P
- tests → set membership
  set comparisons
  set cardinality
- intersect → nested query 'in'
  except → " " 'not in'

- 'some' clause
- 'all' clause → existential quantification

$(5 < \text{all} \begin{bmatrix} 0 \\ 5 \\ 6 \end{bmatrix}) = \text{false}$

- exists return true if the argument subquery is nonempty.
- correlation name → var. s in outer query
- correlation subquery → inner query
- $X - Y = \emptyset \iff X \subseteq Y$
- with ⇒ provides a way of defining a temp. relation whose def^n is only available only to the query in which the 'with' clause occurs
- scalar subquery is one in which is used where a single value is expected

- delete from _____
  where dept = 'finance'
- insert into course
  values ('__', '__', '__', '__', 4);
- update relation
  set salaray = 5 × salary
  where salary > 1000;

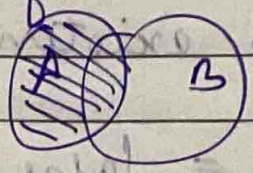
  ] the order is imp.t.

3.3 Intermediate SQL | 2

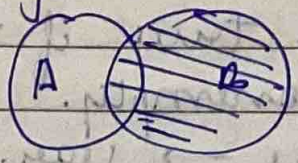
- cross join
- inner join

- outer join — doesn't loose any info.; uses null values

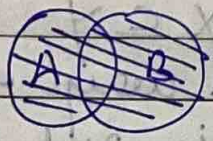
- natural left outer join



- natural right outer join



- natural full outer join



- view → provides a mechanism to hide certain data from the view of certain users
- any ~~view~~ relation that is not of the view for all.
- create view as <query exp^n>
where <query exp^n>

Eg. create view faculty as
select ID, name, dept name
from instructor

now, select name
from faculty
where dept-name = 'Biology';

- → depends directly
→ depend on
→ depends on itself (recursive)
repeat] otherwise
until] loop won't terminate
- materialize a view → then maintain it.

CLASSTIME | Page No.
Date / /

L 3.4 Intermediate SQL | 3

- transacⁿ → unit of work, atomic
  ↳ commit work & roll back work
- by default each statement gets committed
- integrity constraints guard against accident damage to database.
- 1° key can't be null, but candidate key can be null
- check (P) → P is predicate
- referential integrity — it is btw 2 tables, foreign key.
- with cascading, you can define the acⁿ that the database engine takes when a user tries to delete or update a key.
- there could be integrity constraint violation during transacⁿ → solve using 'references'
- data — 2005-7-27
  time — 09:00:30
  timestamp — date + time
  interval
- create index studentID_index on student (ID)] (log n)
- authorizaⁿ —
  ↳ read, insert, update, delete
  ↳ to modify
  index, resources, alteration, drop
- 'grant' statement is used to confer authorizaⁿ
- 'revoke' statement to take back authorizaⁿ
- create role instructor
  grant instructor to Amit

CLASSTIME | Page No.
Date / /

L3.5 Adv. SQL

- Predicate logic / calculus is extension to propositional
- Tuple relational & domain relational calculus are based on predicate calculus → predicate and quantifiers

$$P(x) \equiv x > 3$$
$$\Rightarrow P(5) = T$$
$$P(2) = F$$

- Universal Quantification - mathematical statements sometimes assert that a prop. is true for all values of variable in a particular domain, called the domain of discourse.
- Existential Quantification -
- TRC → non-procedural query lang.

$$\{t \mid P(t)\}$$

Eg. $\{t.Fname \mid Student(t) \wedge t.age > 21\}$

Week-4

L4.1 Formal Relational Query Lang / 1

- relational → σ (select) ; π (project) , ∪ , − , ×
  ρ (rename)
- select ⇒ $\sigma_p(r) = \{t \mid t \in r \text{ & } p(t)\}$
- project ⇒ π : id
- $r \cup s = \{t \mid t \in r \text{ or } t \in s\}$ → union
- difference ⇒ −
- intersection → $r \cap s$
- $r \times s = \{t\,q \mid t \in r \text{ & } q \in s\}$ → cartesian
- division ⇒ $R(Z) \div S(X)$ ⇒ where $X \subseteq Z$
  $Y = Z - X$ $(Z = X \cup Y)$

$$r \div s = \pi_{R-S}(r) - \pi_{R-S}((\pi_{R-S}(r) \times s) - \pi_{R-S,S}(r))$$

L4.2 Part-2

- predicate logic
  $p \vee q \mid \neg p)$ $p <^{T}_{F}$ $q <^{T}_{F}$

$p(x)$ $x > 3$ — predicate

$P(5) \Rightarrow T$
$P(2) \Rightarrow F$ — proposition

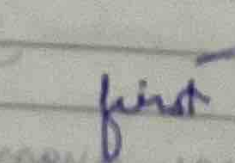
- predicate → use quantifiers
  ① universal
  ② existential
- Eg. $\{t.Fname \mid Student(t) \wedge t.age > 21\}$
- safety of expressions — it is possible to write tuple calculus expression that generate a relations

- domain relational calc.

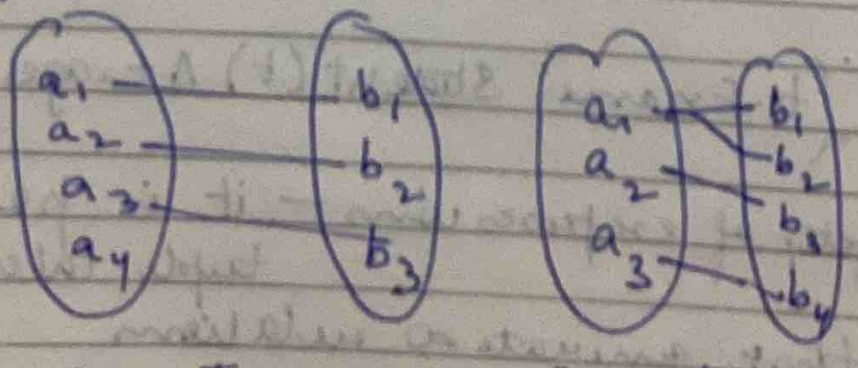
$$\{\langle x_1 \dots x_n \rangle \mid \langle x_1, \dots x_n \in r \wedge b = 17\}$$

L4.3 Entity Relationship Model / 1

- abstraction provides a tool to handle dealing complexity by chunking info.
- 1100101001001
  Octal ⇒ (110)(010)(101)(001) = 6257
  Hex ⇒ (1100)(1010)(1001) = CA9
- logical model → business decision
  → CS decision
- ER Model
  ↳ association among multiple distinguishable entities
  that exist in enterprise
- ER data model → attributes, entity sets, relationship sets
- composite attributes



- relationship set is a mathematical rela' among n ≥ 2 entities.

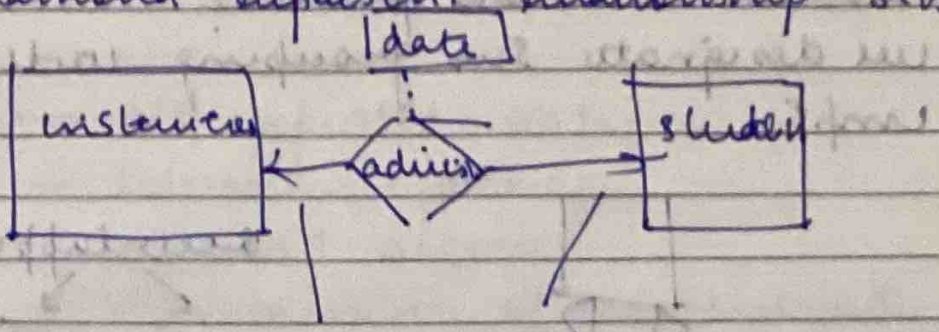


one-to-one one-to-many

CLASSTIME | PAGE NO.
DATE / /

L4.4 ER /2

- diamond represent relationship sets



cardinality constraints

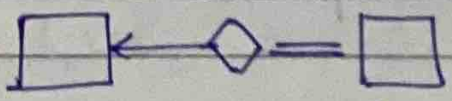
- ID
  name
    first
    middle
    last

complex attributes

{ } → multi valued

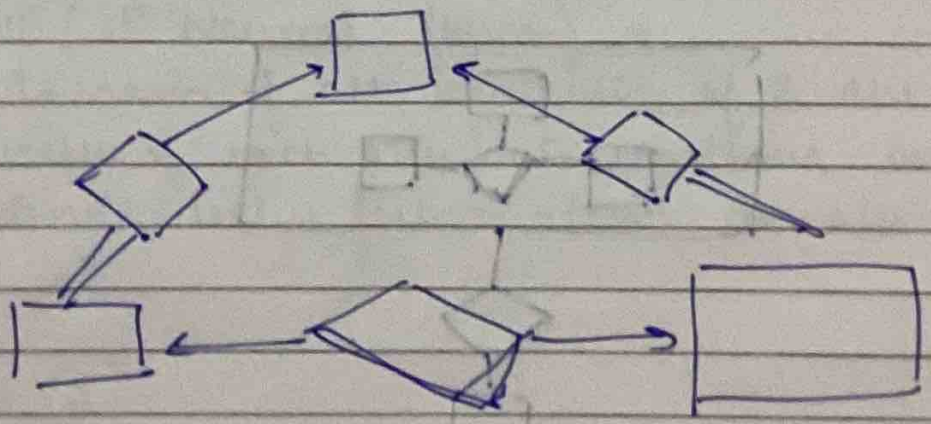
age() → func^n → has to be computed

- Rel^n to relational schema



weak entity

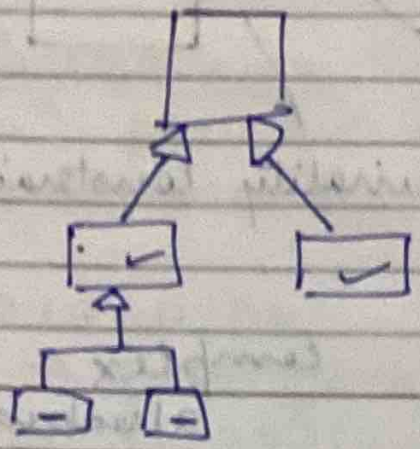
-



many-to-one & one-to-many

L 4.5 EE/2

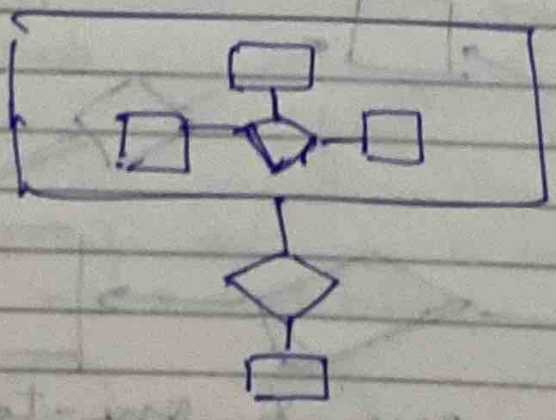
- top down decision process
  ↳ we designate sub grouping within an ent



overlapping

disjoint

- bottom up design process
- aggrega"
  - treat relationship as an abstract entity
  - allow relationship btw. relationship
  -



- binary vs/ non-binary relationships

CLASSTIME | Page No.
Date / /

Week 5

LS.1 Relational DB design-1

- good relational design
  - real-world state.
  - all expected data
  - redundant storage
  - efficient access
  - maintainence of data integrity
  - clean, consistent and easy
- good schema
- redundancy - having multiple copies of same data in DB
- anomaly - insertion, deletion, update
- redundancy ⇒ anomaly
  dependancy ⇒ redundancy
  good decomposi^n ⇒ minimiz^n of dependency
  normaliz^n ⇒ good decomposi^n
- Lossless join decomposi^n

$$R_1 \cup R_2 = R, \quad R_1 \cap R_2 \neq \emptyset$$

$$r_1 \bowtie r_2 = r \quad \forall r$$

- INF (1st Normal form) -
  - domain of all attributes of R are atomic
  - value of each attribute contains only a single value from that domain

LS.2 Part-2

- theory for good rel la^{n}s
  - functional dependencies
  - multivalued dependences
  - other dependencies
- functional —

$\alpha \subseteq R$ and $\beta \subseteq R$

FD

$\alpha \rightarrow \beta$

iff. any tuples t_1 & t_2 of α agree on α, they also agree on β

$$t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$$

- K is a superkey for R iff $K \rightarrow R$
- Armstrong axioms
  - Reflexivity ⇒ $\beta \subseteq \alpha$ then $\alpha \rightarrow \beta$
  - Augmenta" ⇒ $\alpha \Rightarrow \beta$ then $\gamma\alpha \rightarrow \gamma\beta$
  - Transitivity ⇒ $\alpha \rightarrow \beta$ & $\beta \rightarrow \gamma \Rightarrow \alpha \rightarrow \beta$

  → they are sound and complete

$F = \{A \rightarrow B, B \rightarrow C\}$

$F^+ = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$

L 4.3 Part - 3