Lecture 2: Part I

Integrity Rules

Integrity Rules: Applicable to a **Particular** Database

- E.G. An employee's salary must be in the range $30,000 to $100,000.
- See section 8.6 Elmasri & Navathe.
- Supported in the SQL 2 standard and available in most relational systems today.
- Not discussed now.

Integrity Rules: Applicable to **All** Databases

- Key constraint
- Entity integrity
- Referential integrity

Foreign Keys - a Definition

EMP(EMPNO, NAME, DNO, JOB, MGR, SAL, COMMISSION)
DEPT(DNO, DNAME, LOC)

DNO in EMP - should be allowed only if that *DNO* appears as primary key in relation DEPT.

MGR in EMP - should be allowed only if that ***MGR*** appears as primary key in relation EMP.

The attributes DNO and MGR called **foreign keys** in relation EMP.

Def. Foreign Key, FK, an attribute (or combination of attributes in one Relation Schema, R_1, such that:

- Attributes in FK of R_1 have same domain as the primary key attributes, PK, of relation schema R_1.
- A value of FK in a tuple t_1 of R_1 either occurs
  - As a value of PK of some tuple t_2 of R_2,
    - i.e. T_1 [FK] = t_2 [PK]
  - Or is ***null***.

The Three Integrity Rules

- Key constraint

  Candidate keys are specified for each relation schema.

  Candidate key values must be ***unique*** for every tuple in any relation instance of that schema.

The Three Integrity Rules, Continued

- Entity integrity:
  - No attribute participating in the ***primary key*** is allowed to accept **null values.**

Justification: we must be able to identify each tuple uniquely.

The Three Integrity Rules, Continued

- Referential integrity:
  - If relation schema R_1 includes a foreign key, FK, matching the primary key of relation schema R_2, then a value of FK in a tuple t_1 of R_1 must either
    - Be equal to the value of PK in some tuple t_2 of R_2,

      i.e. T_1 [FK] = t_2 [PK], or
    - Be wholly ***null*** (i.e. Each attribute value participating in that FK value must be null).

Note: R_1 and R_2 need not be ***distinct***.

Justification: if some tuple t_1 references some tuple t_2, then tuple t_2 must exist.

Implications of the Three Integrity Rules

What should happen if an **operation** on the database is about to cause the **violation** of one of the **integrity rules**?
i.e. about to put the database into an "illegal" state.

- System could ***reject*** the operation.
- System could ***accept*** the operation, but perform some additional operations so as to reach a new legal state.

Example: Delete DEPT Where DNO = D2

What should happen if there are employees in DEPT with DNO = D2?

- Operation could be **rejected**.
- Employees in DEPT D2 could also be **deleted**, i.e. Cascade the deletion.

Choice should be up to the "user", the database designer.

Notes

- Not included in Codd's original definition of relations.
- Not supported in initial commercial relational database systems.
- Now supported in all major relational database systems.
- Actually introduced in network model.

Foreign Key With ***Null*** Value.

- May an employee ***not*** be assigned to a department?
  - Probably
- May an employee ***not*** have a manager?
  - Let's assume no.

Deleting Target of a Foreign Key Reference

Example: Delete department where there are still employees in the department.

Restricted - delete is ***restricted*** to case where there are no employees in the department.

Cascades - delete ***cascades*** to remove employees also.

Nullifies - foreign key is set to ***null***,
i.e. Employees are no longer in any department.
Then department is deleted.

Updating Target of a Foreign Key Reference.

Example: Change department number (DNO) D50 to D75.

Restricted - update is ***restricted*** to case where there are no employees in the department.

Cascades - update ***cascades*** to change the department numbers of all employees in department D50.

Nullifies - foreign key,
i.e. DNO in EMP is set to ***null***.
Department DNO is updated.

Inserting into Relation With the Foreign Key.

Example: Insert a new employee into department D10, but there is no tuple in the DEPT relation for department D10.

Restricted - insert is ***restricted*** to case where there are already tuples for the corresponding department.

Cascades - if done interactively, user could be asked to enter tuple for department D10.

Nullifies - cannot be done, since user has specified the department number.

Lecture 2: Part II

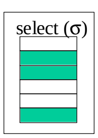
Relational Algebra

Relational Algebra

Operators

- ***Set*** operators
  - Union
  - Intersection
  - Difference
  - Cartesian product
- ***Relational*** operators
  - Selection
  - Projection
  - Join
  - Division

Basic Relational Operators in Pictures



A	B
a1	b1
a2	b1
a3	b2
a4	b2

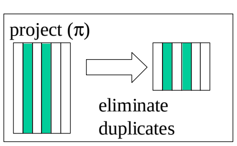
C	D
b1	c1
b2	c2
b2	c3
b3	c3

(natural) join (*)

A	B	D
a1	b1	c1
a2	b1	c1
a3	b2	c2
a3	b2	c3
a4	b2	c2
a4	b2	c3

equijoin

A	B	C	D
a1	b1	b1	c1
a2	b1	b1	c1
a3	b2	b2	c2
a3	b2	b2	c3
a4	b2	b2	c2
a4	b2	b2	c3



Examples (1): Select (σ) and Project (π)

EMP(EMPNO, NAME, DNO, JOB, MGR, SAL, COMMISSION)
DEPT(DNO, DNAME, LOC)

Q1. Find the subset of employees who are in dept. 50.

$\sigma_{DNO = 50}$ (EMP)

Q2. Find the subset of employees who are in dept. 25, 47 or 53.

$\sigma_{DNO = 25 \text{ or } DNO = 47 \text{ or } DNO = 53}$ (EMP)

Q3. List names and jobs of all employees.

$\pi_{NAME, JOB}$ (EMP)

Examples (2): More Complex Select (σ) and Project (π)

EMP(EMPNO, NAME, DNO, JOB, MGR, SAL, COMMISSION)
DEPT(DNO, DNAME, LOC)

Q4. List names and jobs of employees in departments 25, 47 or 53.

$\pi_{NAME, JOB}$ ($\sigma_{DNO = 25 \text{ or } DNO = 47 \text{ or } DNO = 53}$ (EMP))

D_EMPS $\leftarrow$ $\sigma_{DNO = 25 \text{ or } DNO = 47 \text{ or } DNO = 53}$ (EMP)
RESULT $\leftarrow$ $\pi_{NAME, JOB}$ (D_EMP)

Examples (3): Union ($\cup$) and Difference (-)

EMP(EMPNO, NAME, DNO, JOB, MGR, SAL, COMMISSION)
DEPT(DNO, DNAME, LOC)

Q5. List the numbers of those departments which have an employee named ‘Smith’ or are located in ‘Columbus’.

π_{DNO} ($\sigma_{NAME = \text{‘Smith’}}$ (EMP))
$\cup$
π_{DNO} ($\sigma_{LOC = \text{‘Columbus’}}$ (DEPT))

Q6. List the DNO for departments that have no employees.

π_{DNO} (DEPT) - π_{DNO} (EMP)

Examples (4): Join: Natural (*) and Theta (⋈)

EMP(<u>EMPNO</u>, NAME, DNO, JOB, MGR, SAL, COMMISSION)
DEPT(<u>DNO</u>, DNAME, LOC)

Q7. List the names of all employees and the locations of their departments.

$\pi_{\text{NAME, LOC}}$ (EMP * DEPT)) ** Natural join*

$\pi_{\text{NAME, LOC}}$ (EMP ⋈ $_{\text{EMP.DNO = DEPT.DNO}}$ (DEPT))

⋈ Theta-join (θ-Join)
$\theta \; \varepsilon \{\geq, <, =, >, \leq, \neq\}$

Natural join (*)

$Q \leftarrow R_1 *_{\text{(list1), (list2)}} R_2$
list_i has attributes from relation R_i
list_1 names go in Q

Theta (θ) join (⋈)

$Q \leftarrow R_1 \bowtie_{\text{<join condition>}} R_2$
where <**join condition**> is of the form
<condition> AND <condition> AND … AND <condition>
where each **condition** is of the form
$A_i \; \theta \; B_j$, A_i is an attribute of R_1, B_j is an attribute of R_2,
θ (theta) ε ($<, \leq, >, \geq, =, \neq$)

e.g. For each employee, those making higher salary.
EMP1 ⋈ $_{\text{EMP1.SAL < EMP2.SAL}}$ EMP2

Examples (5): Select, project and join

EMP(<u>EMPNO</u>, NAME, DNO, JOB, MGR, SAL, COMMISSION)
DEPT(<u>DNO</u>, DNAME, LOC)

Q8. Find the names of employees who work in Ann Arbor.

π_{NAME} (EMP * ($\sigma_{\text{LOC = 'Ann Arbor'}}$(DEPT)) *Natural join **

π_{NAME} (EMP ⋈ $_{\text{EMP.DNO = DEPT.DNO}}$ ($\sigma_{\text{LOC = 'Ann Arbor'}}$(DEPT)) *equi-join ⋈ when θ is '='*

Examples (6): Not Equal (≠) in Select

EMP(EMPNO, NAME, DNO, JOB, MGR, SAL, COMMISSION)
DEPT(DNO, DNAME, LOC)

Q9. Find the names of employees who ***do not*** work in Ann Arbor.

π_{NAME} (EMP * ($\sigma_{LOC \neq \text{'Ann Arbor'}}$(DEPT)) *Natural join* *

π_{NAME} (EMP ⋈$_{EMP.DNO = DEPT.DNO}$ ($\sigma_{LOC \neq \text{'Ann Arbor'}}$(DEPT)) *(θ) Join* ⋈

Examples (7): A "classic", equi-join as product (×) and select (σ)

EMP(EMPNO, NAME, DNO, JOB, MGR, SAL, COMMISSION)
EMP(EMPNO, NAME, DNO, JOB, MGR, SAL, COMMISSION)

Q10. For each employee whose salary exceeds his or her manager's salary, list the employee's name and the manager's name.

X ← EMP Y ← EMP

$\pi_{X.NAME, Y.NAME}$ ($\sigma_{X.SAL > Y.SAL}$ (X ⋈$_{Y.EMPNO = X.MGR}$ Y))

$\pi_{X.NAME, Y.NAME}$ ($\sigma_{X.SAL > Y.SAL \text{ and } Y.EMPNO = X.MGR}$ (X × Y))

Examples (8): A new kind of query

EMP(EMPNO, NAME, DNO, JOB, MGR, SAL, COMMISSION)
DEPT(DNO, DNAME, LOC)

Q11. Find the numbers of those departments that have employees who can do ***some*** job that is done by an employee in department D3.

Q12. Find the numbers of those departments that have employees who can do ***all*** the jobs that are done by an employee in department D3.

Examples (8): continued

Q11. Find the numbers of those departments that have employees who can do ***some*** job that is done by an employee in department D3. Answer: D1 and D2

EMP(EMPNO, … ,	DNO,	JOB,	. . .)
100	D3	electrician	
200	D3	plumber	
300	D3	electrician	
400	D1	electrician	
500	D1	plumber	
600	D1	carpenter	
700	D2	electrician	
800	D2	carpenter	
900	D2	electrician	

Examples (8): continued

Q12. Find the numbers of those departments that have employees who can do ***all*** the jobs that are done by an employee in department D3. Answer: D1 but not D2

EMP(EMPNO, … ,	DNO,	JOB,	. . .)
100	D3	electrician	
200	D3	plumber	
300	D3	electrician	
400	D1	electrician	
500	D1	plumber	
600	D1	carpenter	
700	D2	electrician	
800	D2	carpenter	
900	D2	electrician	

Examples (8): continued (exists, ∃)

Q11. Find the numbers of those departments that have employees who can do ***some*** job that is done by an employee in department D3. Answer: D1 and D2

EMP(EMPNO, … ,	DNO,	JOB,	. . .)
100	D3	electrician	
200	D3	plumber	

EMP(EMPNO, … ,	DNO,	JOB,	. . .)
500	D1	plumber	
900	D2	electrician	

$\pi_{DNO} (\pi_{JOB} (\sigma_{DNO = D3} (EMP)) * EMP)$

What about department D3?

Examples (8): continued

Q12. Find the numbers of those departments that have employees who can do ***all*** the jobs that are done by an employee in department D3. Answer: D1, but not D2

EMP(<u>EMPNO</u>, … ,	DNO,	JOB,	. . .)
100	D3	electrician	
200	D3	plumber	
300	D3	electrician	
400	D1	electrician	
500	D1	plumber	
600	D1	carpenter	
700	D2	electrician	
800	D2	carpenter	
900	D2	electrician	

Examples (9): divide (÷)

SP(S#, P#)

S#	P#
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P2
S3	P2
S4	P1
S4	P2
S4	P3

÷ P(P#): P1, P2, P3 ⇒ SP ÷ P (S#): S1, S4

Examples (9): divide (÷)

Dividend ÷ Divisor = Quotient

(degree m + n) (degree n) (degree m)

SP(S#, P#)

S#	P#
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P2
S3	P2
S4	P1
S4	P2
S4	P3

÷ P(P#): P1, P2, P3 ⇒ SP ÷ P (S#): S1, S4

÷ P(P#): P1, P2 ⇒ SP ÷ P (S#): S1, S2, S4

÷ P(P#): P1 ⇒ SP ÷ P (S#): S1, S2, S4

Examples (8): Divide (÷), for all (∀)

Q12. Find the numbers of those departments that have employees who can do ***all*** the jobs that are done by an employee in department D3. Answer: D1, but not D2

EMP(

EMPNO	DNO	JOB	. . .)
100	D3	electrician	
200	D3	plumber	
300	D3	electrician	
400	D1	electrician	
500	D1	plumber	
600	D1	carpenter	
700	D2	electrician	
800	D2	carpenter	
900	D2	electrician	

D3JOBS $\leftarrow \pi_{JOB}(\sigma_{DNO = D3}(EMP))$

DEPT_JOBS $\leftarrow \pi_{DNO, JOB}(EMP)$

GOOD_DEPTS $\leftarrow$ DEPT_JOBS ÷ D3JOBS

$(\pi_{DNO, JOB}(EMP)) \div (\pi_{JOB}(\sigma_{DNO = D3}(EMP)))$

Examples (10): Functions and groups

Q13. Find the average salary of all employees.

EMP(

EMPNO	DNO	SAL,...)
100	D3	66,000
200	D3	55,000
300	D3	66,000
400	D1	65,000
500	D1	55,000
600	D1	60,000
700	D2	66,000
800	D2	60,000
900	D2	66,000

⇒ 62,000

$\mathcal{F}_{AVERAGE\ SAL}(EMP)$

Examples (10): Functions and groups continued

Q14. List the departments (DNO) and the average salary of each.

EMP(

EMPNO	DNO	SAL,...)
100	D3	66,000
200	D3	55,000
300	D3	66,000
400	D1	66,000
500	D1	55,000
600	D1	60,000
700	D2	66,000
800	D2	60,000
900	D2	66,000

⇒

DNO	AVG SAL
D3	62,333
D1	60,333
D2	64,000

$_{DNO}\mathcal{F}_{AVERAGE\ SAL}(EMP)$

Examples (11): Outer joins

P(<u>P#</u>, PNAME, CITY)

P#	PNAME	CITY
P1	Nut	London
P3	Screw	Rome

S(<u>S#</u>, SNAME, CITY)

S#	SNAME	CITY
S1	Smith	London
S5	Adams	Athens

Q15a. For each part list all suppliers in the same city.

(P#	PNAME	S#	SNAME	CITY)
P1	Nut	S1	Smith	London

$P \bowtie_{P.CITY = S.CITY} S$

natural join (inner join)

Q15b. For each part list all suppliers in the same city.
For a part with no supplier in the city, list null.

(P#	PNAME	S#	SNAME	CITY)
P1	Nut	S1	Smith	London
P3	Screw	?	?	Rome

$P ⟕_{P.CITY = S.CITY} S$

left outer join

Examples (11): Outer joins continued

P(<u>P#</u>, PNAME, CITY)

P#	PNAME	CITY
P1	Nut	London
P3	Screw	Rome

S(<u>S#</u>, SNAME, CITY)

S#	SNAME	CITY
S1	Smith	London
S5	Adams	Athens

Q15c. For each part list all suppliers in the same city.
For a supplier with no part in the city, list null.

(P#	PNAME	S#	SNAME	CITY)
P1	Nut	S1	Smith	London
?	?	S5	Adams	Athens

$P ⟖_{P.CITY = S.CITY} S$

right outer join

Q15d. For each part list all suppliers in the same city.
For a part with no supplier in the city, list null.
For a supplier with no part in the city, list null.

(P#	PNAME	S#	SNAME	CITY)
P1	Nut	S1	Smith	London
P3	Screw	?	?	Rome
?	?	S5	Adams	Athens

$P ⟗_{P.CITY = S.CITY} S$

outer join

Examples (12): Recursive closure

EMP(<u>EMPNO</u>, MGR,...)

EMPNO	MGR
100	700
200	700
300	700
400	800
500	800
600	800
700	900
800	900
900	900

Q 16. List all the superiors of EMPNO 100.
700
900

Q 17. List all those supervised by EMPNO 800.
600

Can’t express these queries? WHY?

Relational Completeness

- Select, project, union, difference, cartesian product sufficient to express
  - Join, division and intersection
- Is there any new language this powerful?
  - Must be able to translate ***any*** relational algebra statement to the new language.