Database Management Systems

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Lecture 7, 3 November 2023

Constraints on a Single Relation

- **not null**
- **primary key**
- **unique**
- **check** (P), where P is a predicate

Not Null Constraints

- **not null**
  - Declare *name* and *budget* to be **not null**

    name **varchar**(20) **not null**
    budget **numeric**(12,2) **not null**

Unique Constraints

- **unique** ($A_1, A_2, \ldots, A_m$)
  - The unique specification states that the attributes $A_1, A_2, \ldots, A_m$ form a candidate key.
  - Candidate keys are permitted to be null (in contrast to primary keys).

The check clause

- The **check** (P) clause specifies a predicate P that must be satisfied by every tuple in a relation.
- Example: ensure that semester is one of fall, winter, spring or summer

```
create table section
   (course_id varchar (8),
    sec_id varchar (8),
    semester varchar (6),
    year numeric (4,0),
    building varchar (15),
    room_number varchar (7),
    time slot id varchar (4),
    primary key (course_id, sec_id, semester, year),
    check (semester in ('Fall', 'Winter', 'Spring', 'Summer')))
```

Referential Integrity

- Ensures that a value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relation.
  - Example: If "Biology" is a department name appearing in one of the tuples in the *instructor* relation, then there exists a tuple in the *department* relation for "Biology".
- Let A be a set of attributes. Let R and S be two relations that contain attributes A and where A is the primary key of S. A is said to be a **foreign key** of R if for any values of A appearing in R these values also appear in S.

Referential Integrity (Cont.)

- Foreign *keys can be* specified as part of the SQL **create table** statement

  foreign key (*dept_name*) **references** *department*

- By default, a foreign key references the primary-key attributes of the referenced table.
- SQL allows a list of attributes of the referenced relation to be specified explicitly.

  foreign key (*dept_name*) **references** *department* (*dept_name*)

Cascading Actions in Referential Integrity

- When a referential-integrity constraint is violated, the normal procedure is to reject the action that caused the violation.
- An alternative, in case of delete or update is to cascade

```
create table course (
    (…
    dept_name varchar(20),
    foreign key (dept_name) references department
        on delete cascade
        on update cascade,   // propagate the change
    . . .)
```

- Instead of cascade we can use :
  - **set null**,
  - **set default**

Built-in Data Types in SQL

- **date:** Dates, containing a (4 digit) year, month and date
  - Example: **date** '2005-7-27'
- **time:** Time of day, in hours, minutes and seconds.
  - Example: **time** '09:00:30' **time** '09:00:30.75'
- **timestamp:** date plus time of day
  - Example: **timestamp** '2005-7-27 09:00:30.75'
- **interval:** period of time
  - Example: interval '1' day
  - Subtracting a date/time/timestamp value from another gives an interval value
  - Interval values can be added to date/time/timestamp values

Advanced SQL

- Many other features
  - Transactions
  - Assertions and triggers
  - . . .

Advanced SQL

- Many other features
  - Transactions
  - Assertions and triggers
  - ...
- Can call SQL from other programming languages
  - Almost every language has library functions to invoke SQL
  - Transfer data between online forms and databases
  - ...

Security — SQL injection attacks

- User input can be malicious commands to corrupt database
- Always validate data entered in a form before passing on to SQL

- User input can be malicious commands to corrupt database
- Always validate data entered in a form before passing on to SQL

Relational database design

- Set of attributes that one needs to keep track of

Relational database design

- Set of attributes that one needs to keep track of
- Why not combine into a single table?

Relational database design

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Relational database design

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

- Combine these into a single table?

Relational database design

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Relational database design

- Redundant storage

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Relational database design

- Redundant storage
- Maintaining consistency
  - Updates
  - Inserts and deletes

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Decomposition and information

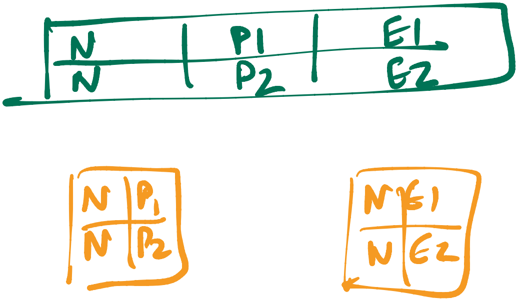
- (customer_name,regd_phone,regd_email)

Decomposition and information

- (customer_name,regd_phone,regd_email)
- Decompose as (customer_name,regd_phone) and (customer_name,regd_email)

Decomposition and information

- (customer_name,regd_phone,regd_email)
- Decompose as (customer_name,regd_phone) and (customer_name,regd_email)
- Name is not unique — loss of information



Decomposition and information

- (customer_name,regd_phone,regd_email)
- Decompose as (customer_name,regd_phone) and (customer_name,regd_email)
- Name is not unique — loss of information
- Recombining decomposed relation should not add tuples

Decomposition and information

- `(customer_name,regd_phone,regd_email)`
- Decompose as `(customer_name,regd_phone)` and `(customer_name,regd_email)`
- Name is not unique — loss of information
- Recombining decomposed relation should not add tuples
- Lossless decomposition
  - Decompose R as R_1 and R_2
  - Want $R = R_1 \bowtie R_2$

Functional dependencies

- $A_1, A_2, \ldots, A_k \rightarrow B_1, B_2, \ldots B_m$
  - LHS atributes uniquely fix RHS attributes
  - Must hold for every instance — semantic property of attributes

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Functional dependencies

- $A_1, A_2, \ldots, A_k \rightarrow B_1, B_2, \ldots B_m$
  - LHS atributes uniquely fix RHS attributes
  - Must hold for every instance — semantic property of attributes
- Need not correspond to superkeys
  - `dept_name` $\rightarrow$ `building`
  - `dept_name` $\rightarrow$ `budget`

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

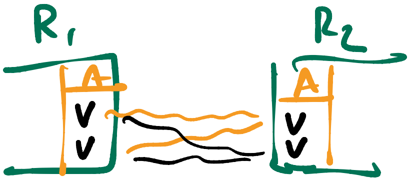
Functional dependencies

- $A_1, A_2, \ldots, A_k \rightarrow B_1, B_2, \ldots B_m$
  - LHS atributes uniquely fix RHS attributes
  - Must hold for every instance — semantic property of attributes
- Need not correspond to superkeys
  - `dept_name` $\rightarrow$ `building`
  - `dept_name` $\rightarrow$ `budget`
- Use to identify sources of redundancy, guide decomposition

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Lossless decomposition and functional dependencies

- Decompose R as R_1 and R_2



Lossless decomposition and functional dependencies

- Decompose R as R_1 and R_2
- Decomposition is lossless if at least one of the following functional dependencies hold
  - $R_1 \cap R_2 \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$

Lossless decomposition and functional dependencies

- Decompose R as R_1 and R_2
- Decomposition is lossless if at least one of the following functional dependencies hold
  - $R_1 \cap R_2 \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$
- Decompose `Instructor-Department` as `Instructor` and `Department`
  - `Instructor` $\cap$ `Department` is `dept_name`
  - `dept_name` is primary key for `Department`

Lossless decomposition and functional dependencies

- Decompose R as R_1 and R_2
- Decomposition is lossless if at least one of the following functional dependencies hold
  - $R_1 \cap R_2 \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$

] To achieve lossless decomp

- Decompose `Instructor-Department` as `Instructor` and `Department`
  - `Instructor` $\cap$ `Department` is `dept_name`
  - `dept_name` is primary key for `Department`
- In general need to compute all implied dependencies
  - From $A \rightarrow B$ and $B \rightarrow C$, conclude that $A \rightarrow C$
- Closure of a set of dependencies F — denoted F^+

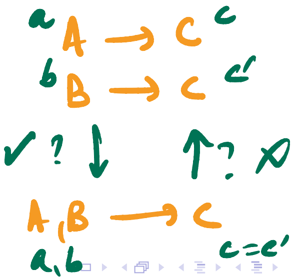
$A \rightarrow B, \; A \rightarrow C$

$\Downarrow \quad \uparrow ?$

$A \rightarrow B, C$

Lossless decomposition and functional dependencies

- Decompose R as R_1 and R_2
- Decomposition is lossless if at least one of the following functional dependencies hold
  - $R_1 \cap R_2 \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$
- Decompose `Instructor-Department` as `Instructor` and `Department`
  - `Instructor` $\cap$ `Department` is `dept_name`
  - `dept_name` is primary key for `Department`
- In general need to compute all implied dependencies
  - From $A \rightarrow B$ and $B \rightarrow C$, conclude that $A \rightarrow C$
- Closure of a set of dependencies F — denoted F^+



Computing the closure of a set of attributes

- Given $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$ and B, does $A_1, A_2, \ldots, A_k \rightarrow B$?

Given some functional dependences

Computing the closure of a set of attributes

- Given $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$ and B, does $A_1, A_2, \ldots, A_k \rightarrow B$?
- Iterative algorithm — check if B is in closure $\mathcal{A}^+$

Initialize $\mathcal{A}^+$ to $\{A_1, A_2, \ldots, A_k\}$
repeat
  for each $\beta \rightarrow \gamma$ in F
    if $\beta \subseteq \mathcal{A}^+$, add γ to $\mathcal{A}^+$
  end
until no change in $\mathcal{A}^+$

$\mathcal{A}^+ = \mathcal{A} \cup \{B_1, B_2, \ldots, B_m\}$

$\mathcal{A} \rightarrow B_i$ for each i

$\mathcal{A} \rightarrow \mathcal{B}$ for any $\mathcal{B} \subseteq \{B_1, \ldots, B_m\}$

$A_3, A_7 \rightarrow C$

Add C to $\mathcal{A}^+$

$A_8, C \rightarrow B$

Add B to $\mathcal{A}^+$

$\vdots$

$A_9, B \rightarrow C$

Normal forms

- Criteria to determine if the collection of tables is “good”

Normal forms

- Criteria to determine if the collection of tables is “good”
- Normalization — decompose tables till they achieve a normal form

Normal forms

- Criteria to determine if the collection of tables is “good”
- Normalization — decompose tables till they achieve a normal form
- Guided by functional dependencies

Boyce-Codd Normal Form (BCNF)

- Relational schema R, set of functional dependencies F

Boyce-Codd Normal Form (BCNF)

- Relational schema R, set of functional dependencies F
- Write α, β to represent sequences of attributes $A_1, A_2, \ldots, A_k$, $B_1, B_2, \ldots, B_m$

Boyce-Codd Normal Form (BCNF)

- Relational schema R, set of functional dependencies F
- Write α, β to represent sequences of attributes $A_1, A_2, \ldots, A_k$, $B_1, B_2, \ldots, B_m$
- R is in BCNF if, for every $\alpha \rightarrow \beta \in F^+$ one of the following holds
  - $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
  - α is a superkey for R

closure

Boyce-Codd Normal Form (BCNF)

- Relational schema R, set of functional dependencies F
- Write α, β to represent sequences of attributes $A_1, A_2, \ldots, A_k, B_1, B_2, \ldots, B_m$
- R is in BCNF if, for every $\alpha \rightarrow \beta \in F^+$, one of the following holds
  - $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
  - α is a superkey for R
- `InstructorDepartment(ID,name,salary,dept_name,building,budget` not in BCNF

Boyce-Codd Normal Form (BCNF)

- Relational schema R, set of functional dependencies F
- Write α, β to represent sequences of attributes $A_1, A_2, \ldots, A_k$, $B_1, B_2, \ldots, B_m$
- R is in BCNF if, for every $\alpha \rightarrow \beta \in F^+$, one of the following holds
  - $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
  - α is a superkey for R
- `InstructorDepartment(ID,name,salary,dept_name,building,budget` not in BCNF
- `Instructor(ID,name,dept_name,salary)` and `Department(dept_name,building,budget)` are in BCNF

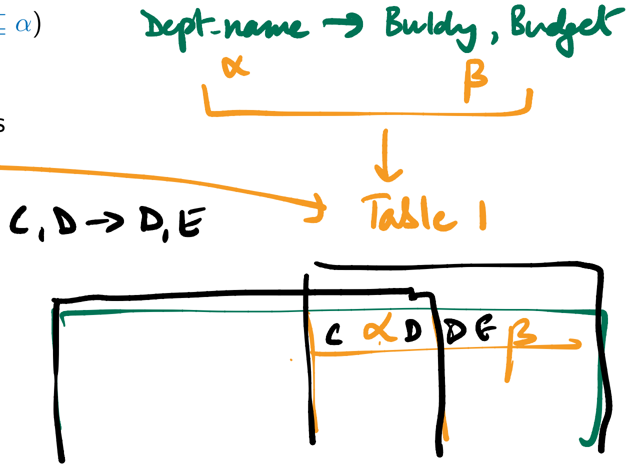
Achieving BCNF

- $\alpha \rightarrow \beta \in F^+$ is a BCNF violation for R if neither of the following holds
  - $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
  - α is a superkey for R

Achieving BCNF

- $\alpha \rightarrow \beta \in F^+$ is a BCNF violation for R if neither of the following holds
  - $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
  - α is a superkey for R
- To fix this, decompose R as
  - $\alpha \cup \beta$
  - $R \setminus (\beta \setminus \alpha)$

R \ β



Achieving BCNF

- $\alpha \rightarrow \beta \in F^+$ is a BCNF violation for R if neither of the following holds
  - $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
  - α is a superkey for R
- To fix this, decompose R as
  - $\alpha \cup \beta$
  - $R \setminus (\beta \setminus \alpha)$
- Example: `dept_name` $\rightarrow$ `building,budget` is a BCNF violation for `InstructorDepartment(ID,name,salary,dept_name,building,budget`

Achieving BCNF

- $\alpha \rightarrow \beta \in F^+$ is a BCNF violation for R if neither of the following holds
  - $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
  - α is a superkey for R
- To fix this, decompose R as
  - $\alpha \cup \beta$
  - $R \setminus (\beta \setminus \alpha)$
- Example: `dept_name` $\rightarrow$ `building,budget` is a BCNF violation for `InstructorDepartment(ID,name,salary,dept_name,building,budget`
- Decompose as
  - `Department(dept_name,building,budget)` $\alpha \cup \beta$
  - `Instructor(ID,name,dept_name,salary)` $R \setminus \beta$

Dependency preservation

- `Advisor(student_id,faculty_id,dept_name)`
- Each faculty member is in only one department
- Students can be across multiple departments
- Each student has at most one advisor in each department

Dependency preservation

- Advisor(student_id,faculty_id,dept_name) — BCNF violation
- Each faculty member is in only one department
- Students can be across multiple departments
- Each student has at most one advisor in each department
- BCNF decomposition is (student_id,faculty_id), (faculty_id,dept_name)

$R - \beta$ $\alpha \cup \beta$

fac_id → dept_name

sid, dept → fid

Dependency preservation

- Advisor(student_id,faculty_id,dept_name)
- Each faculty member is in only one department
- Students can be across multiple departments
- Each student has at most one advisor in each department
- BCNF decomposition is (student_id,faculty_id), (faculty_id,dept_name)
- Functional dependencies
  - faculty_id → dept_name
  - student_id,dept_name → faculty_id

requires join

Dependency preservation

- Advisor(student_id,faculty_id,dept_name)

Key

- Each faculty member is in only one department
- Students can be across multiple departments
- Each student has at most one advisor in each department
- BCNF decomposition is (student_id,faculty_id), (faculty_id,dept_name)
- Functional dependencies
  - faculty_id $\rightarrow$ dept_name
  - student_id,dept_name $\rightarrow$ faculty_id
- Need join to check second dependency

Cannot "locally check"

Third normal form (3NF)

- R is in 3NF if, for every $\alpha \rightarrow \beta \in F^+$, one of the following holds
  - $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
  - α is a superkey for R
  - Each attribute A in $\beta \setminus \alpha$ is contained in some candidate key for R

Third normal form (3NF)

- R is in 3NF if, for every $\alpha \rightarrow \beta \in F^+$, one of the following holds
  - $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$
  - α is a superkey for R
  - Each attribute A in $\beta \setminus \alpha$ is contained in some candidate key for R
- BCNF is a stricter condition than 3NF

R in BCNF

⇒ R in 3NF

⇐ ✗

Third normal form (3NF)

- R is in 3NF if, for every $\alpha \rightarrow \beta \in F^+$, one of the following holds
  - $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$
  - α is a superkey for R
  - Each attribute A in $\beta \setminus \alpha$ is contained in some candidate key for R
- BCNF is a stricter condition than 3NF
- Priorities
  - Lossless decomposition — Cannot be compromised
  - BCNF — lack of redundancy
  - Dependency preservation