RDBMS and SQL

Madhavan Mukund

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

Lecture 8, 23 October 2025

Relational database design

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

IPL 2024 data

IPL 2025

Ball by ball

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

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

Instructor

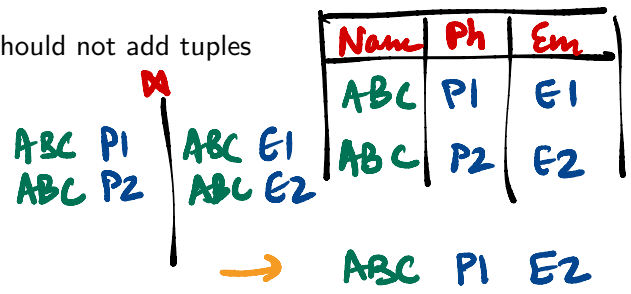
ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	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

dept

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 $R = R_1 \bowtie R_2$

with overlap



Functional dependencies

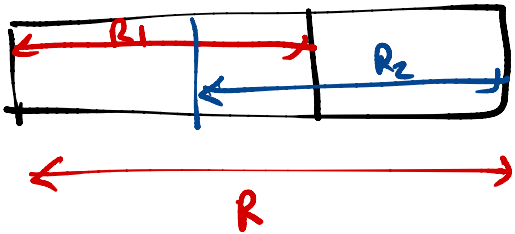
- $A_1, A_2, \ldots, A_k \rightarrow B_1, B_2, \ldots B_m$
  - LHS atributes uniquely fix RHS attributes
  - Must hold for every instance — semantic property of attributes
- Need not correspond to superkeys
  - `dept_name` $\rightarrow$ `building`
  - `dept_name` $\rightarrow$ `budget`
- Use to identify 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
- Decomposition is lossless if at least one of the following functional dependencies hold
  - $R_1 \cap R_2 \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$

sufficient but not necessary



ABC	P1	E1
ABC	P2	E2
ABC	P1	E2
ABC	P2	E1

(Name, Phone) ⋈ (Name, Email)

= Original table

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
  - $R_1 \cap R_2 \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$
- 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^+

Given

To be computed

Computing the closure of a set of attributes

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

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 $\mathcal{A}^+$

All B such that $\mathcal{A} \rightarrow B$

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 $\mathcal{A}^+$

$A_1 \rightarrow A_1$

A_1
$(a_1, \ldots, a_k)$
a_1

Must terminate because finite attributes

$A \longrightarrow A^+$

What we want is $F \longrightarrow F^+$

Take any $X, Y \subseteq$ attributes, is $X \to Y \in F^+$?

Check if $Y \subseteq X^+$

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

in a relation R

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

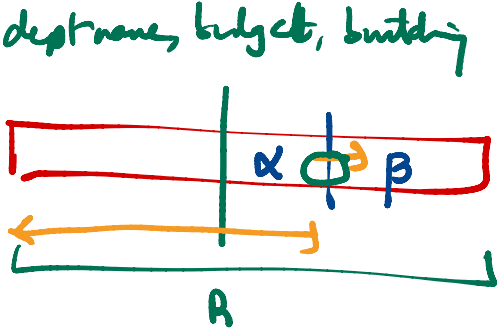
- $\alpha \to \beta \in F^+$ is a BCNF violation for R if neither of the following holds
  - $\alpha \to \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
  - α is a superkey for R
- To fix this, decompose R as
  - $\alpha \cup \beta$
  - $R \setminus (\beta \setminus \alpha)$

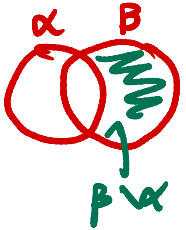
$A_1 A_2 \to A_2 B$ ✓ ✗

$\beta \setminus \alpha \Rightarrow \beta$

β'' if $\alpha \cap \beta = \phi$

$R \setminus \beta$





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)`
  - `Instructor(ID,name,dept_name,salary)`

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)
- Each faculty member is in only one department
- Students can be across multiple departments
- Each student has at most one advisor in each department
- Functional dependencies
  - faculty_id $\rightarrow$ dept_name
  - student_id,dept_name $\rightarrow$ faculty_id

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
- Functional dependencies
  - faculty_id $\rightarrow$ dept_name
  - student_id,dept_name $\rightarrow$ faculty_id
- BCNF decomposition is (student_id,faculty_id), (faculty_id,dept_name)

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
- Functional dependencies
  - faculty_id $\rightarrow$ dept_name
  - student_id,dept_name $\rightarrow$ faculty_id
- BCNF decomposition is (student_id,faculty_id), (faculty_id,dept_name)
- Need join to check dependency student_id,dept_name $\rightarrow$ faculty_id

Dependency preservation, formally

- Given a set of dependencies F and a decomposition of R as $R_1, R_2, \ldots, R_k$

Dependency preservation, formally

- Given a set of dependencies F and a decomposition of R as $R_1, R_2, \ldots, R_k$
- Can locally check a dependency $\alpha \rightarrow \beta$ in R_i if $\alpha \cup \beta \subseteq R_i$

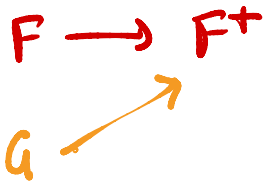
Dependency preservation, formally

- Given a set of dependencies F and a decomposition of R as $R_1, R_2, \ldots, R_k$
- Can locally check a depenency $\alpha \rightarrow \beta$ in R_i if $\alpha \cup \beta \subseteq R_i$
- Let F_i be set of dependencies in F^+ ocally checkable in R_i

Dependency preservation, formally

- Given a set of dependencies F and a decomposition of R as $R_1, R_2, \ldots, R_k$
- Can locally check a depenency $\alpha \rightarrow \beta$ in R_i if $\alpha \cup \beta \subseteq R_i$
- Let F_i be set of dependencies in F^+ locally checkable in R_i
- Let $G = F_1 \cup F_2 \cup \cdots \cup F_k$. Is $G^+ = F^+$?

$F_1 \cup F_2 \cdots \cup F_k = F^+$

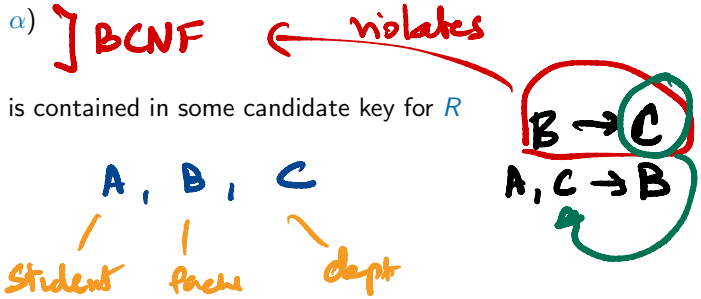


Dependency preservation, formally

- Given a set of dependencies F and a decomposition of R as $R_1, R_2, \ldots, R_k$
- Can locally check a depenency $\alpha \rightarrow \beta$ in R_i if $\alpha \cup \beta \subseteq R_i$
- Let F_i be set of dependencies in F^+ locally checkable in R_i
- Let $G = F_1 \cup F_2 \cup \cdots \cup F_k$. Is $G^+ = F^+$?
- How do we compute F_i for each R_i?
  - Let R_i have attributes $A_1, A_2, \ldots, A_m$
  - For each subset α of $A_1, A_2, \ldots, A_m$, compute α^+ with respect to F^+
  - For each $B \in \alpha^+ \cap \{A_1, A_2, \ldots, A_m\}$, add $\alpha \rightarrow B$ to R_i

Third normal form (3NF)

- R is in 3NF if, for every $\alpha \to \beta \in F^+$, one of the following holds
  - $\alpha \to \beta$ 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



Why 3NF ? What about 1NF & 2NF?

All attributes are atomic

early form of BCNF

Third normal form (3NF)

- R is in 3NF if, for every $\alpha \to \beta \in F^+$, one of the following holds
  - $\alpha \to \beta$ 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

Third normal form (3NF)

- R is in 3NF if, for every $\alpha \to \beta \in F^+$, one of the following holds
  - $\alpha \to \beta$ 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
  - BCNF
  - Dependency preservation

Beyond functional dependencies

- Suppose we collect emergency contact details for each students — phone and email
  - At least two emergency contacts of each type

Beyond functional dependencies

- Suppose we collect emergency contact details for each students — phone and email
  - At least two emergency contacts of each type
- Consider a table `Emergency(student_id,phone,email)`
  - Two phone numbers and two emails will generate four rows

Beyond functional dependencies

- Suppose we collect emergency contact details for each students — phone and email
  - At least two emergency contacts of each type
- Consider a table `Emergency(student_id,phone,email)`
  - Two phone numbers and two emails will generate four rows
- This redundancy cannot be explained in terms of functional dependencies

Beyond functional dependencies

- Suppose we collect emergency contact details for each students — phone and email
  - At least two emergency contacts of each type
- Consider a table `Emergency(student_id,phone,email)`
  - Two phone numbers and two emails will generate four rows
- This redundancy cannot be explained in terms of functional dependencies
- Multivalued dependency — closure under swaps

More like an independence

Beyond functional dependencies

- Suppose we collect emergency contact details for each students — phone and email
  - At least two emergency contacts of each type
- Consider a table `Emergency(student_id,phone,email)`
  - Two phone numbers and two emails will generate four rows
- This redundancy cannot be explained in terms of functional dependencies
- Multivalued dependency — closure under swaps
- 4NF — flags "independence" & splits

Practical matters

- Validating functional dependencies

Practical matters

- Validating functional dependencies
- Redundancy vs computing joins — materialized views