Outline

- Mechanisms and definitions to work with FDs:
  - Closures, candidate keys, canonical covers etc...
  - Armstrong axioms
- Decompositions:
  - Loss-less decompositions, Dependency-preserving decompositions
- BCNF:
  - How to achieve a BCNF schema
- **BCNF may not preserve dependencies**
- 3NF: Solves the above problem
- BCNF allows for redundancy
- 4NF: Solves the above problem

---

**BCNF may not preserve dependencies**

- \( R = \{J, K, L\} \)
- \( F = \{JK \rightarrow L, L \rightarrow K\} \)
  - Candidate keys?
    - \( JK, JL \)
    - \( JK \rightarrow L \) defines key, \( L \rightarrow K \) does not
      - \( R1 = \{LK\}, \) deps = \( L \rightarrow K \)
      - \( R2 = \{JL\}, \) no deps
        - lost \( JK \rightarrow L \)
  - \( R \) is not in BCNF
BCNF may not preserve dependencies

- Not always possible to find a dependency-preserving decomposition that is in BCNF.
- PTIME to determine if there exists a dependency-preserving decomposition in BCNF
  - in size of F
- NP-Hard to find one if it exists
- Better results exist if F satisfies certain properties

Prime attributes

- Definition: *Prime attributes*
  - An attribute that is contained in a candidate key for R
- Example 1:
  - \( R = \{A, B, C, D, E, H\} \), \( F = \{A \rightarrow BC, E \rightarrow HA\} \),
  - Candidate keys = \{ED\}
  - Prime attributes: D, E
- Example 2:
  - \( R = \{J, K, L\} \), \( F = \{JK \rightarrow L, L \rightarrow K\} \),
  - Candidate keys = \{JL, JK\}
  - Prime attributes: J, K, L

- Observation/Intuition:
  1. A key has no redundancy (is not repeated in a relation)
  2. A *prime attribute* has limited redundancy
3NF to the rescue

$R$ is in **3NF (3rd Normal Form)** if:

- Given a relation schema $R$ and a set of functional dependencies $F$:
  - if every FD, $\alpha \rightarrow \beta$, is either:
    - Trivial, or
    - $\alpha$ is a superkey of $R$, or
    - All attributes in $(\beta - \alpha)$ are prime

**Why is 3NF good?**

- Lossless
- Preserves dependencies.
- Limited redundancy

3NF and Redundancy

**Why does redundancy arise?**

- Given a FD, $\alpha \rightarrow \beta$, if $\alpha$ is repeated $(\beta - \alpha)$ has to be repeated
  - If rule 1 is satisfied, $(\beta - \alpha)$ is empty, so not a problem.
  - If rule 2 is satisfied, $\alpha$ can’t be repeated, so this doesn’t happen either
  - If not, rule 3 says $(\beta - \alpha)$ must contain only prime attributes
    - This limits the redundancy somewhat.

- 3NF relaxes BCNF by allowing some (hopefully limited) redundancy
- Why is 3NF good?
  - There always exists a dependency-preserving lossless decomposition in 3NF.
Decomposing into 3NF

let $F_c$ be a canonical cover for $F$;
i := 0;
for each functional dependency $\alpha \rightarrow \beta$ in $F_c$
i := $i + 1$;
$R_i := \alpha \beta$;
if none of the schemas $R_j$, $j = 1, 2, \ldots, i$ contains a candidate key for $R$then
i := $i + 1$;
$R_i :=$ any candidate key for $R$;
/* Optionally, remove redundant relations */
repeat
if any schema $R_j$ is contained in another schema $R_k$then
/* Delete $R_j */
$R_j := R_i$;
i := $i - 1$;
until no more $R_j$s can be deleted
return $(R_1, R_2, \ldots, R_i)$

Figure 8.12 Dependency-preserving, lossless decomposition into 3NF.

3NF Example

- $(R) = (A, B, C, D, E, F, G, H)$
- Function Dependencies
  - $F = \{A \rightarrow CGH, AD \rightarrow C, DE \rightarrow F, G \rightarrow H\}$
    - $R_1 = \{ACGH\}, R_2 = \{ADC\}, R_3 = \{DEF\}, R_4 = \{GH\}$
    - $R_5 = \{ABDE\}$
  - $F' = \{A \rightarrow CG, AD \rightarrow C, DE \rightarrow F, G \rightarrow H\}$
    - $H$ is extra in $A \rightarrow CGH$
    - $R_1 = \{ACGH\}, R_2 = \{ADC\}, R_3 = \{DEF\}, R_4 = \{GH\}, R_5 = \{ABDE\}$
    - $R_1 = \{ACG\}, R_2 = \{DEF\}, R_3 = \{GH\}$
    - $R_5 = \{ABDE\}$

- Somewhat better if start from canonical cover: $F' =$
  - $\{A \rightarrow CG, AD \rightarrow C, DE \rightarrow F, G \rightarrow H\}$
  - $D$ extra in $AD \rightarrow C$
  - merge $A \rightarrow C$ into $A \rightarrow CG$

- so $F_c = \{A \rightarrow CG, DE \rightarrow F, G \rightarrow H\}$
  - $R_1 = \{ACG\}, R_2 = \{DEF\}, R_3 = \{GH\}$
  - $R_5 = \{ABDE\}$

- Lossless: Each (except $R_4$) has a single FD that defines a key
- Preserves dependencies: each carried through a single subrelation
Outline

- Mechanisms and definitions to work with FDs
  - Closures, candidate keys, canonical covers etc...
  - Armstrong axioms
- Decompositions
  - Loss-less decompositions, Dependency-preserving decompositions
- BCNF
  - How to achieve a BCNF schema
- BCNF may not preserve dependencies
- 3NF: Solves the above problem
- BCNF allows for redundancy
- 4NF: Solves the above problem

---

BCNF and redundancy

<table>
<thead>
<tr>
<th>MovieTitle</th>
<th>MovieYear</th>
<th>StarName</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star wars</td>
<td>1977</td>
<td>Harrison Ford</td>
<td>Address 1, LA</td>
</tr>
<tr>
<td>Star wars</td>
<td>1977</td>
<td>Harrison Ford</td>
<td>Address 2, FL</td>
</tr>
<tr>
<td>Indiana Jones</td>
<td>198x</td>
<td>Harrison Ford</td>
<td>Address 1, LA</td>
</tr>
<tr>
<td>Indiana Jones</td>
<td>198x</td>
<td>Harrison Ford</td>
<td>Address 2, FL</td>
</tr>
<tr>
<td>Witness</td>
<td>19xx</td>
<td>Harrison Ford</td>
<td>Address 1, LA</td>
</tr>
<tr>
<td>Witness</td>
<td>19xx</td>
<td>Harrison Ford</td>
<td>Address 2, FL</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

No non-trivial FDs, so the schema is trivially in BCNF (and 3NF)
But lots of redundancy.

4NF using *multi-valued dependencies (MVDs)*, read up if interested
Multi-valued Dependencies

- The redundancy is because of *multi-valued dependencies*
- Denoted:
  - starname →→ address
  - starname →→ movietitle, moviyear
- Does not happen if schema is constructed from an E/R diagram
- Functional dependencies are a special case of multi-valued dependencies

Multi-valued Dependencies

- FDs *rule out* certain tuples
- MVDs *require* tuples of certain form

- Read up if interested.
- Not on test.
4NF

- Similar to BCNF, except with MVDs instead of FDs.

- Given a relation schema \( R \), and a set of multi-valued dependencies \( F \), if every MVD, \( A \rightarrow\rightarrow B \), is either:
  1. Trivial, or
  2. \( A \) is a superkey of \( R \)

Then, \( R \) is in 4NF (4th Normal Form)

- 4NF \( \rightarrow \) BCNF \( \rightarrow \) 3NF \( \rightarrow \) 2NF \( \rightarrow \) 1NF:
  - If a schema is in 4NF, it is in BCNF.
  - If a schema is in BCNF, it is in 3NF.

- Other way round is not necessarily true.

### Comparing the normal forms

<table>
<thead>
<tr>
<th></th>
<th>3NF</th>
<th>BCNF</th>
<th>4NF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eliminates redundancy because of FD’s</td>
<td>Mostly</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Eliminates redundancy because of MVD’s</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Preserves FDs</td>
<td>Yes.</td>
<td>Maybe</td>
<td>Maybe</td>
</tr>
<tr>
<td>Preserves MVDs</td>
<td>Maybe</td>
<td>Maybe</td>
<td>Maybe</td>
</tr>
</tbody>
</table>

4NF is typically desired and achieved.

A good E/R diagram won’t generate non-4NF relations at all

Choice between 3NF and BCNF is up to the designer
Database design process

- So three ways to come up with a schema:
  1. Using E/R diagram
     - If good, then little normalization is needed
     - Tends to generate 4NF designs
  2. A universal relation $R$ that contains all attributes.
     - Called universal relation approach
     - Note that MVDs will be needed in this case
  3. An *ad hoc* schema that is then normalized
     - MVDs may be needed in this case

Recap

- What about 1\textsuperscript{st} and 2\textsuperscript{nd} normal forms?
  - 1NF:
    - Essentially says that no set-valued attributes allowed
    - Formally, a domain is called *atomic* if the elements of the domain are considered indivisible
    - A schema is in 1NF if the domains of all attributes are atomic
    - We assumed 1NF throughout the discussion
      - Non 1NF is just not a good idea
  - 2NF:
    - Mainly historic interest
    - See Exercise 7.15 in the book
Recap

- We would like our relation schemas to:
  - *Not allow potential redundancy* because of FDs or MVDs
  - Be *dependency-preserving*:
    - Make it easy to check for dependencies
    - Since they are a form of integrity constraints
- Functional Dependencies/Multi-valued Dependencies
  - Domain knowledge about the data properties
- Normal forms
  - Defines the rules that schemas must follow
  - 4NF is preferred, but 3NF is sometimes used instead

Recap

- Denormalization
  - After doing the normalization, we may have too many tables
  - We may *denormalize* for performance reasons
    - Too many tables → too many joins during queries
  - Another option is to use *views* instead
    - If a specific set of tables is joined often, create a view on the join
- More advanced normal forms
  - project-join normal form (PJNF or 5NF)
  - domain-key normal form
  - Rarely used in practice
Q2
0.1 Points

What is the result of the boolean expression:

\(((\text{NULL} = 20) \text{ or } (10 = 10)) \text{ and } ((\text{NULL} = 10) \text{ is unknown})\)

- True
- False
- Unknown

Q3
0.1 Points

What is the result of:

\(((\text{False and Unknown) is Unknown}) \text{ or } ((\text{True or Unknown) is Unknown})\)

- True
- False
- Unknown

---

Q10
0.1 Points

What is the result of:

```sql
with t as (select A from R),
    t2 as (select * from R where B = 10)
(select * from t) intersect (select A from t2)
```

- (a, b, c)
- [a, a, b, c, c]
- (b, c, c)
- [a, b]
Quiz3

Q11
0.2 Points

Consider two create table statements:

```sql
create table R (a integer primary key);
create table S (b integer primary key,
    c integer references R(a) on update cascade);
```

What will happen when a tuple in R is updated or deleted (the answer may be different for the two)?

Q12
0.2 Points

Are the following two queries equivalent? Why or Why not? Assume R.a is an integer attribute.

1. select * from R where R.a > 1;
2. (select * from R) except (select * from R where R.a <= 1);

Q13

1. $\pi_B(\sigma_{B=c}(R))$
2. $\pi_C(R) - \pi_{R.C}(\sigma_{R.C < R.C}(R \times \rho_{R_1}(R)))$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>a</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>b</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>α</td>
<td>c</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>α</td>
<td>a</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>γ</td>
<td>c</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>γ</td>
<td>a</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>b</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>α</td>
<td>c</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>β</td>
<td>c</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

(1) is [c]. More interestingly, if the projection was to A, the answer would be α, β, and γ. (2)'s answer [1].
Quiz 3

Q14 0.2 Points
Suppose we have three relations r(A, B), s(B, C), and t(B, D) with all attributes declared as not null. Consider the expression:

(r natural left outer join s) natural left outer join t

Give instances of relations r, s and t such that in the result of the expression, attribute C has a null value but attribute D has a non-null value.

EXPLANATION:
r = \{(x, y)\}
s = \{(2, z)\}
t = \{(y, a)\}
gives \{(x, y, null)\}

Q15 0.2 Points
For three relations R(A, B), S(B, C), T(C, D), write relational algebra expressions to generate the following relations:

1. Q1(A, D) where R and S are joined on condition R.B > S.B, and S and T have a natural join.
2. Q2(A, C) to find all (A, C) pairs such that R.B = S.B, and S.C does not have a matching tuple in T.

In both cases, use only the basic relational operations.

EXPLANATION:
Q1 \leftarrow \pi_{A,D}(R \bowtie S \bowtie T)
Q2 \leftarrow \pi_{A,C}(T \bowtie \sigma_{R.B=S.B}(R \bowtie S))

Note that natural joins are both commutative and associative.

Quiz 4

Q1 10 Points
The following two questions are on a relation:

racetimes(athlete, time). The first query below uses the rank() function to rank the athletes by time (lower time is better). The second query below is a rewrite of the query shown in slides to calculate ranks using basic SQL. Here we count the number of tuples with a lower or equal time and use that as the rank. However, the two queries do not always return the same result. Explain with an example.

```sql
select name, rank() over (order by time asc) as a_rank
from racetimes
order by a_rank;

select name, (select count(*)
from racetimes b
where b.time <= a.time) as a_rank
from racetimes a
order by a_rank;
```

EXPLANATION:
With ties the latter skips numbers before the two tied, instead of after. Try with ".”
Also true that the two approaches behave differently with nulls.

create table racetimes(name varchar, time int);
insert into racetimes values (’peter’, 3);
insert into racetimes values (’paul’, 2);
insert into racetimes values (’john’, 2);
select name, rank() over (order by time asc) as a_rank
from racetimes order by a_rank;

name | a_rank
-----
peter | 1
paul  | 2
john  | 2
bubba | 4
(4 rows)

select name, (select count(*)
from racetimes b
where b.time <= a.time) as a_rank
from racetimes a
order by a_rank;

name | a_rank
-----
peter | 1
paul  | 3
john  | 3
bubba | 4
(4 rows)
**Q6**

10 Points

Convert the following two E/R diagrams into a minimal relational schema, i.e., a relational schema that has fewest possible relations that captures the E/R model. Indicate any primary/discriminant keys (in all these questions). Assume total participation in both.

**R**

attr1

attr2

role1

role2

**EXPLANATION**

(bold face means primary key attribute)

R(attr1, attr2)

rel(role1_attr1, role2_attr1)

---

**Q7**

10 Points

Convert the following E/R diagram into a minimal relational schema, i.e., a relational schema that has fewest possible relations that captures the E/R model. Assume total participation of S in both relationship sets.

**R**

attr1

attr2

**S**

attr3

attr4

rel1

rel2

**EXPLANATION**

(bold face means primary key attribute)

R(attr1, attr2), S(attr3, attr4, attr1)

rel2(attr1, attr3)