Outline

BCNF decomposition

1) Check whether chosen FD violates BCNF
2) Use any FD that violates BCNF to decompose.

View construction and query processing

1) From vertically partitioned tables
2) From horizontally partitioned tables
We call an attribute that determines all other attributes in a schema to be a superkey.

If it is the smallest set of attributes (in terms of cardinality) that does this we call that set a minimal key or just key.
Closure Algorithm

Goal:
We want everything that an attribute/set of attributes determine

Observation:
- If we have $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$.
- So really, $A \rightarrow B$ and $C$.
- Formal notation is $\{A\}^+ = \{A, B, C\}$.
- **Since the closure of $A$ is all attributes, $A$ is a key.**

**Repeat until** $X$ doesn’t change **do**:
- **if** $B_1, ..., B_n \rightarrow C$ is a FD and $B_1, ..., B_n$ are all in $X$ **then** add $C$ to $X$. 

## Conceptual Design

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>PhoneNumber</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-1234</td>
<td>Seattle</td>
</tr>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-6543</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>908-555-2121</td>
<td>Westfield</td>
</tr>
</tbody>
</table>
Anomalies:

- Redundancy = repeat data
- Update anomalies = what if Fred moves to “Bellevue”? 
- Deletion anomalies = what if Joe deletes his phone number?
The BCNF (Boyce-Codd Normal Form) ---- A relation $R$ is in BCNF if every set of attributes is either a superkey or its closure is the same set
BCNF Decomposition Algorithm

Normalize(R)

find X s.t.: $X \neq X^+$ and $X^+ \neq \{\text{all attributes}\}$

if (not found) then $R$ is in BCNF

let $Y = X^+ - X$; \hspace{1em} $Z = \{\text{all attributes}\} - X^+$

decompose $R$ into $R_1(X \cup Y)$ and $R_2(X \cup Z)$

Normalize($R_1$); Normalize($R_2$);
Example

The relation is $R (A, B, C, D, E)$

FDs: $A \rightarrow E$, $BC \rightarrow A$, and $DE \rightarrow B$

Question: Decompose R into BCNF.
Notice that $\{A\}^+ = \{A,E\}$, which violates the BCNF condition.

We split $R$ to $R_1(A,E)$ and $R_2(A,B,C,D)$.

$R_1$ satisfies BCNF now, but $R_2$ does not because: $\{B,C\}^+ = \{B,C,A\}$.

Notice that there is no $E$ in $R_2$ table so we don't need to consider the FD $DE \rightarrow B$.

Split $R_2$ to: $R_{21}(B,C,A)$ and $R_{22}(B,C,D)$
Consider the relation $R(A,B,C,D,E)$

FDs: $\{ AB \rightarrow C, \ BC \rightarrow D, \ AD \rightarrow E \}$

$S_1 = \Pi_{ABC}(R)$, $S_2 = \Pi_{BCD}(R)$, $S_3 = \Pi_{ADE}(R)$

We need to show that $R = S_1 \bowtie S_2 \bowtie S_3$
Vertical Partitioning

<table>
<thead>
<tr>
<th>Resumes</th>
<th>SSN</th>
<th>Name</th>
<th>Address</th>
<th>Resume</th>
<th>Picture</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>234234</td>
<td>Mary</td>
<td>Houston</td>
<td>Doc1…</td>
<td>JPG1…</td>
</tr>
<tr>
<td></td>
<td>345345</td>
<td>Sue</td>
<td>Seattle</td>
<td>Doc2…</td>
<td>JPG2…</td>
</tr>
<tr>
<td></td>
<td>345343</td>
<td>Joan</td>
<td>Seattle</td>
<td>Doc3…</td>
<td>JPG3…</td>
</tr>
<tr>
<td></td>
<td>432432</td>
<td>Ann</td>
<td>Portland</td>
<td>Doc4…</td>
<td>JPG4…</td>
</tr>
</tbody>
</table>

T1

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>234234</td>
<td>Mary</td>
<td>Houston</td>
</tr>
<tr>
<td>345345</td>
<td>Sue</td>
<td>Seattle</td>
</tr>
<tr>
<td>…</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

T2

<table>
<thead>
<tr>
<th>SSN</th>
<th>Resume</th>
</tr>
</thead>
<tbody>
<tr>
<td>234234</td>
<td>Doc1…</td>
</tr>
<tr>
<td>345345</td>
<td>Doc2…</td>
</tr>
</tbody>
</table>

T3

<table>
<thead>
<tr>
<th>SSN</th>
<th>Picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>234234</td>
<td>JPG1…</td>
</tr>
<tr>
<td>345345</td>
<td>JPG2…</td>
</tr>
</tbody>
</table>

T2.SSN is a key and a foreign key to T1.SSN. Same for T3.SSN.
Vertical Partitioning

CREATE VIEW Resumes AS
SELECT T1.ssn, T1.name, T1.address,
     T2.resume, T3.picture
FROM   T1, T2, T3
WHERE  T1.ssn=T2.ssn AND T1.ssn=T3.ssn

SELECT address
FROM   Resumes
WHERE  name = 'Sue'
Vertical Partitioning

Original query:

```
SELECT T1.address
FROM T1, T2, T3
WHERE T1.name = 'Sue'
    AND T1.SSN = T2.SSN
    AND T1.SSN = T3.SSN
```

Final query:

```
SELECT T1.address
FROM T1
WHERE T1.name = 'Sue'
```
Vertical Partitioning Applications

● **Advantages**
  ○ Speeds up queries that touch only a small fraction of columns
  ○ Single column can be compressed effectively, reducing disk I/O

● **Disadvantages**
  ○ Updates are very expensive!
  ○ Need many joins to access many columns
  ○ Repeated key columns add overhead
Horizontal Partitioning

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>234234</td>
<td>Mary</td>
<td>Houston</td>
</tr>
<tr>
<td>345345</td>
<td>Sue</td>
<td>Seattle</td>
</tr>
<tr>
<td>345343</td>
<td>Joan</td>
<td>Seattle</td>
</tr>
<tr>
<td>234234</td>
<td>Ann</td>
<td>Portland</td>
</tr>
<tr>
<td>--</td>
<td>Frank</td>
<td>Calgary</td>
</tr>
<tr>
<td>--</td>
<td>Jean</td>
<td>Montreal</td>
</tr>
</tbody>
</table>

CustomersInHouston

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>234234</td>
<td>Mary</td>
<td>Houston</td>
</tr>
</tbody>
</table>

CustomersInSeattle

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>345345</td>
<td>Sue</td>
<td>Seattle</td>
</tr>
<tr>
<td>345343</td>
<td>Joan</td>
<td>Seattle</td>
</tr>
</tbody>
</table>
CREATE VIEW Customers AS
(SELECT SSN, name, ‘Houston’ as city
FROM CustomersInHouston)
UNION ALL
(SELECT SSN, name, ‘Seattle’ as city
FROM CustomersInSeattle)
UNION ALL

...
Horizontal Partitioning

```
SELECT name
FROM Customers
WHERE city = 'Seattle'
```

```
SELECT name
FROM CustomersInSeattle
```
Horizontal Partitioning Applications

- **Performance optimization**
  - Especially for data warehousing
  - E.g., one partition per month
  - E.g., archived applications and active applications

- **Distributed and parallel databases**