Logistics

Gradience
No action items for now
Detailed instructions coming shortly
First quiz to be released next Monday (April 4)

Exams
Open book, open laptop
Midterm in class on Wednesday, April 27
Final starting 12:15pm on Friday, June 3

Logistics

Polls
Lecture pace: just right (86%)
Printed handouts: mostly no (77%) → will keep bringing some
Review session: yes (52%), focus more on implementation (60%)
CA office hours

Distributed Database Design

Bottom-up approach
Multi-databases
Individual
No design issues!

Top-down approach
Start with a blank slate
Similar to centralized DB design
New distribution issues
New Issues

Fragmentation
How to split the data?

Allocation
Where should each fragment go?

→ Not independent, but will cover them separately

Example

Employee relation
E(id, name, salary, location, ...)

40% of queries
Q_A: select *
from E
where location = A and ...

40% of queries
Q_B: select *
from E
where location = B and ...

Motivation: two sites

Example

Employee relation
E(id, name, salary, location, ...)

Fragmentation
Straightforward...
Fragmentation

Straightforward...

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>location</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tom</td>
<td>A</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>Ann</td>
<td>B</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>Ben</td>
<td>A</td>
<td>21</td>
</tr>
</tbody>
</table>

\[ F = \{ F_1, F_2 \} \]

\[ F_1 = \sigma_{location=A}E \]
\[ F_2 = \sigma_{location=B}E \]

\[ E \]

\[ F \]

\[ F_1 \]

\[ F_2 \]

\[ \Rightarrow \text{Primary horizontal fragmentation} \]
Horizontal Partitioning Techniques

Round robin partitioning
Hash partitioning
Range partitioning

Round Robin Partitioning
Distributes data evenly
Good for scanning full relation
Not good for point or range queries

Hash Partitioning
Distributes data evenly if hash function is good
Good for point queries on key and joins
Not good for range queries and point queries not on key

Range Partitioning
Good for some range queries on partition attribute
Need to select good vector to avoid data/execution skew
Example 1:
\[ F = \{ F_1, F_2 \} \]
\[ F_1 = \sigma_{\text{salary}<10} E \]
\[ F_2 = \sigma_{\text{salary}>20} E \]
→ Problem: some tuples are lost \((10 \leq \text{salary} < 20)\)

Example 2:
\[ F = \{ F_3, F_4 \} \]
\[ F_3 = \sigma_{\text{salary}<10} E \]
\[ F_4 = \sigma_{\text{salary}>5} E \]
→ Problem: some tuples are duplicated \((5 < \text{salary} < 10)\)
**Good Fragmentations?**
Prefer to deal with replication separately:
First fragment...
\[ F = \{ F_5, F_6, F_7 \} \]
\[ F_5 = \sigma_{\text{salary} \leq 5} \]
\[ F_6 = \sigma_{5 < \text{salary} < 10} \]
\[ F_7 = \sigma_{\text{salary} \geq 10} \]
... then replicate, e.g., \( F_6 \) if necessary (as part of allocation)

**Fragmentation Properties**
\[ R \Rightarrow F = \{ F_1, F_2, \ldots \} \]
Desired properties of horizontal fragmentations:
(1) Completeness \( \forall t \in R, \exists F_i \in F : t \in F_i \)
(2) Disjointness \( \forall t \in F_i, \neg \exists F_j : t \in F_j \land i \neq j \land \{ F_i, F_j \} \subset F \)
(3) Reconstruction \( R = \bigcup_{F_i \subset F} F_i \)

**Fragmentation Properties**
How to ensure completeness and disjointness?
(1) Check manually
\[ F_1 = \sigma_{\text{salary} \leq 5} \]
\[ F_2 = \sigma_{\text{salary} > 5} \]

(2) Automatically generate fragments with such properties
Minterm Predicates

(1) Consider simple predicates used in queries
E.g., salary < 10, salary > 5, location = A, location = B
(2) Generate minterm predicates
(3) Eliminate useless ones

Minterm Predicates

(1) salary < 10 ∧ salary > 5 ∧ location = A ∧ location = B
(2) salary < 10 ∧ salary > 5 ∧ location = A ∧ ~location = B
(3) salary < 10 ∧ salary > 5 ∧ ~location = A ∧ location = B
(4) salary < 10 ∧ ~salary > 5 ∧ location = A ∧ ~location = B
(5) salary < 10 ∧ ~salary > 5 ∧ ~location = A ∧ location = B
(6) salary < 10 ∧ ~salary > 5 ∧ ~location = A ∧ ~location = B
(7) salary < 10 ∧ ~salary > 5 ∧ ~location = A ∧ ~location = B
(8) salary < 10 ∧ ~salary > 5 ∧ ~location = A ∧ ~location = B
Minterm Predicates

(9) \(\neg(sal < 10) \land sal > 5 \land loc = A \land loc = B\)
(10) \(\neg(sal < 10) \land sal > 5 \land loc = A \land \neg(loc = B)\)
(11) \(\neg(sal < 10) \land sal > 5 \land \neg(loc = A) \land loc = B\)
(12) \(\neg(sal < 10) \land sal > 5 \land \neg(loc = A) \land \neg(loc = B)\)
(13) \(\neg(sal < 10) \land \neg(sal < 5) \land loc = A \land loc = B\)
(14) \(\neg(sal < 10) \land \neg(sal < 5) \land \neg(loc = A) \land loc = B\)
(15) \(\neg(sal < 10) \land \neg(sal < 5) \land \neg(loc = A) \land \neg(loc = B)\)
(16) \(\neg(sal < 10) \land \neg(sal < 5) \land \neg(loc = A) \land \neg(loc = B)\)

Minterm Predicates

(9) \(\neg(sal < 10) \land sal > 5 \land loc = A \land loc = B\)
(10) \(\neg(sal < 10) \land sal > 5 \land loc = A \land \neg(loc = B)\)
(11) \(\neg(sal < 10) \land sal > 5 \land \neg(loc = A) \land loc = B\)
(12) \(\neg(sal < 10) \land sal > 5 \land \neg(loc = A) \land \neg(loc = B)\)
(13) \(\neg(sal < 10) \land \neg(sal < 5) \land loc = A \land loc = B\)
(14) \(\neg(sal < 10) \land \neg(sal < 5) \land \neg(loc = A) \land loc = B\)
(15) \(\neg(sal < 10) \land \neg(sal < 5) \land \neg(loc = A) \land \neg(loc = B)\)
(16) \(\neg(sal < 10) \land \neg(sal < 5) \land \neg(loc = A) \land \neg(loc = B)\)

Final fragments:

\(F_2\) \(5 < salary < 10 \land location = A\)
\(F_3\) \(5 < salary < 10 \land location = B\)
\(F_6\) \(salary \leq 5 \land location = A\)
\(F_7\) \(salary \leq 5 \land location = B\)
\(F_{10}\) \(salary \geq 10 \land location = A\)
\(F_{11}\) \(salary \geq 10 \land location = B\)
Minterm Predicates

Elimination of useless fragments depends on application semantics

E.g., if location could be other than A or B, must add fragments:

\[ F_4 \quad 5 < \text{salary} < 10 \land \text{location} \neq A \land \text{location} \neq B \]
\[ F_8 \quad \text{salary} \leq 5 \land \text{location} \neq A \land \text{location} \neq B \]
\[ F_{12} \quad \text{salary} \geq 10 \land \text{location} \neq A \land \text{location} \neq B \]

Why does this technique work?

To illustrate, consider \( p_1, p_2, p_3, p_4 \)

Predicates

\[ p_1 \land p_2 \land p_3 \land p_4 \]
\[ p_1 \land p_2 \land p_3 \land \neg p_4 \]
\[ \vdots \quad \vdots \quad \vdots \quad \vdots \]
\[ \neg p_1 \land \neg p_2 \land \neg p_3 \land \neg p_4 \]

Minterm Predicates

(1) Completeness

\[ \forall t \in R: p_i(t) \text{ is either true or false} \]

Say, \( p_1(t) \) and \( p_2(t) \) are true; \( p_3(t) \) and \( p_4(t) \) are false

Then \( t \) is in the fragment with predicate \( p_1 \land p_2 \land \neg p_3 \land \neg p_4 \)

Minterm Predicates

(2) Disjointness

Say \( t \) is in the fragment with predicate \( p_1 \land p_2 \land \neg p_3 \land \neg p_4 \)

Then \( p_1(t) \) and \( p_2(t) \) are true; \( p_3(t) \) and \( p_4(t) \) are false

There is no other fragment \( t \) could be in
Minterm Predicates

Summary
Given a set of simple predicates \( P = \{ p_1, p_2, \ldots, p_n \} \), minterm predicates are

\[
M = \{ m \mid m = \bigwedge_{1 \leq k \leq n} p_k^* \}
\]

where \( p_k^* \) is either \( p_k \) or \( \neg p_k \)

\( \rightarrow \) Fragments \( \sigma_m R \) for all \( m \in M \) are complete and disjoint

Matching Access Patterns

Another desired fragmentation property

Matching Access Patterns

Earlier example:

\[
E(id, \text{name}, \text{salary}, \text{location}, \ldots)
\]

40% of queries

\[
\begin{align*}
Q_A &: \quad \text{select}^* \\
& \quad \text{from} \ E \\
& \quad \text{where} \ \text{location} = A \ \text{and} \ \ldots
\end{align*}
\]

40% of queries

\[
\begin{align*}
Q_B &: \quad \text{select}^* \\
& \quad \text{from} \ E \\
& \quad \text{where} \ \text{location} = B \ \text{and} \ \ldots
\end{align*}
\]

Matching Access Patterns

Consider the following options:

(1) \( P = \{ \} \)

\( F_1 = \{ E \} \)

(2) \( P = \{ \text{location} = A, \text{location} = B \} \)

\( F_2 = \{ \sigma_{\text{location} = A} E, \sigma_{\text{location} = B} E \} \)

(3) \( P = \{ \text{location} = A, \text{location} = B, \text{salary} < 10 \} \)

\( F_3 = \{ \sigma_{\text{location} = A} \land \text{salary} < 10 E, \sigma_{\text{location} = A} \land \text{salary} \geq 10 E, \sigma_{\text{location} = B} \land \text{salary} < 10 E, \sigma_{\text{location} = B} \land \text{salary} \geq 10 E \} \)
Matching Access Patterns

In other words:

\[ \begin{align*}
F_1 & = \{ \text{select ... location=A ...} \\
F_2 & = \{ \text{select ... location=B ...} \}
\end{align*} \]

\[ F_2 \text{ is good (not } F_1 \text{ and } F_3) \]

Derived Horizontal Fragmentation

Example:

\[ E(\text{id, name, salary, location, ...}) \]
\[ F = \{ F_1, F_2 \} \text{ by location} \]

\[ T(\text{id, task, ...}) \]

Common query for task:
Given employee name, list tasks (s)he works on
### Derived Horizontal Fragmentation

#### Table 1: Employees

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>location</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tom</td>
<td>A</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>Ben</td>
<td>A</td>
<td>21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>location</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Ann</td>
<td>B</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>Max</td>
<td>B</td>
<td>17</td>
</tr>
</tbody>
</table>

#### Table 2: Tasks

<table>
<thead>
<tr>
<th>id</th>
<th>task</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>design</td>
</tr>
<tr>
<td>1</td>
<td>build</td>
</tr>
<tr>
<td>2</td>
<td>advertise</td>
</tr>
<tr>
<td>4</td>
<td>sell</td>
</tr>
</tbody>
</table>

### Derived Horizontal Fragmentation

Let $R, F = \{ F_1, F_2, \ldots \}$ and $S, G = \{ G_1, G_2, \ldots \}$, where $G_i = S \bowtie F_i$.

**Convention**
- $R$ is owner
- $S$ is member

F could be primary or derived.

### Checking completeness and disjointness

Example:

<table>
<thead>
<tr>
<th>id</th>
<th>task</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>test</td>
</tr>
</tbody>
</table>

but no $id = 6$ in $E_1$ or in $E_2$

This $T$ tuple will not be in either $T_1$ or $T_2$. Fragmentation is not complete.
**Derived Horizontal Fragmentation**

To get completeness need to enforce **referential integrity**:

- Join attribute of the member relation
- Each value must be present
- Join attribute of the owner relation

---

**Derived Horizontal Fragmentation**

Example:

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Location</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tom</td>
<td>A</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Location</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ann</td>
<td>B</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Horizontal Fragmentation Summary**

**Types**
- Primary
- Derived

**Properties**
- Completeness
- Disjointness
### Vertical Fragmentation

**Example:**

\[
\begin{array}{ccc}
\text{id} & \text{name} & \text{location} \\
1 & \text{Tom} & \text{A} \\
2 & \text{Ann} & \text{B} \\
3 & \text{Ben} & \text{A} \\
\end{array}
\]

\[
\begin{array}{cc}
\text{id} & \text{salary} \\
1 & 15 \\
2 & 23 \\
3 & 21 \\
\end{array}
\]

\[
E = E_1 \cup E_2
\]

### Vertical Fragmentation

**Properties**

\( R[A] \Rightarrow R_i[A_i] \)

(1) **Completeness**

\( \cup A_i = A \quad \forall i \)

(2) **Disjointness**

\( A_i \cap A_j = \emptyset \quad \forall i, j \quad i \neq j \)

\[
\begin{array}{cc}
\text{id} & \text{location} \quad \text{salary} \\
E_1 & \text{(id, location)} \quad E_2 & \text{(salary)}
\end{array}
\]

\( A = \{ a_1, a_2, ..., a_n \} \) set of attributes

\( A_i \subseteq A \)
Vertical Fragmentation

(2) Disjointness

\[ A_i \cap A_j = \emptyset \quad \forall i,j : i \neq j \]

\[ E_1(id, location, salary) \]

\[ E_2(salary) \]

Not a desirable property (could not reconstruct R)

Vertical Fragmentation

(3) Lossless join

\[ \bowtie R_i = R \quad \forall i \]

One way to achieve lossless join

Repeat key in all fragments

key \subseteq A_i \quad \forall i

Vertical Fragmentation

How do we decide what attributes are grouped with which?

Example:

\[ E = (id, name, location) \]

\[ E = (id, salary) \]

\[ E = (id, name, location, salary) \]

\[ E = (id, name) \]

\[ E = (id, location) \]

\[ E = (id, salary) \]

Attribute Affinity Matrix

\[
\begin{bmatrix}
    a_1 & a_2 & a_3 & a_4 & a_5 \\
    a_1 & - & - & - & - \\
    a_2 & - & - & - & - \\
    a_3 & - & 45 & 48 & - \\
    a_4 & - & 1 & 2 & 0 & - \\
    a_5 & - & 0 & 4 & 75 & - \\
\end{bmatrix}
\]
Attribute Affinity Matrix

<table>
<thead>
<tr>
<th></th>
<th>a₁</th>
<th>a₂</th>
<th>a₃</th>
<th>a₄</th>
<th>a₅</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a₂</td>
<td>50</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a₃</td>
<td>45</td>
<td>48</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a₄</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a₅</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>75</td>
<td>-</td>
</tr>
</tbody>
</table>

R₁[k, a₁, a₂, a₃]  R₂[k, a₄, a₅]

Distributed DB Design Issues

**Fragmentation ✔**
How to split the data?

**Allocation**
Where should each fragment go?

Issues

Where do queries originate?
What is the communication cost?
What is the storage capacity and cost at sites?
Size of fragments?
What is the processing power at sites?
What is the query processing strategy?
How are joins done?
Where are answers collected?

Allocation

Example:
E (id, name, location, salary), F₁=σ_{location=A}E, F₂=σ_{location=B}E

Qₐ: select * from E where location = A and …
Qₐ: select * from E where location = B and …

Where do F₁, F₂ go?

Site A ? Site B
Issues

Do we replicate fragments?
- Cost of updating copies?
- How to provide concurrency control?

Optimization

Best placement of fragments and/or best number of copies to
- Minimize query response time
- Maximize throughput
- Minimize some other cost (e.g., communication)
Subject to constraints
- Available storage
- Available bandwidth, power, etc.
- Other, e.g., keep 90th percentile response time < x

This is an incredibly hard problem

Optimization

Example
Single fragment F

Read cost \sum\limits_{i=1}^{m} \left( t_i \times \min_j C_{ij} \right)

i \quad \text{originating site of request}

\begin{align*}
t_i & \quad \text{read traffic at } S_i \\
C_{ij} & \quad \text{retrieval cost (accessing fragment } F \text{ at } S_j \text{ from } S_i)\
\end{align*}
Write cost scenario

- \( F \) originating site of request
- \( j \) site being updated
- \( x_j \) 0 if \( F \) not stored at \( S_j \)
- \( u_i \) 1 if \( F \) stored at \( S_i \)
- \( c_{ij} \) write traffic at \( S_i \)
- \( c'_{ij} \) write cost (updating \( F \) at \( S_j \) from \( S_i \))

Storage cost

\[ \sum_{i} x_i D_i \]

- \( x_i \) 0 if \( F \) not stored at \( S_i \)
- 1 if \( F \) stored at \( S_i \)
- \( D_i \) storage cost at \( S_i \)
Optimization

Target function

\[
\min \left\{ \sum_{j=1}^{m} t_j \times \min C_{ij} + \sum_{j=1}^{m} x_j \times u_i \times C'_{ij} \right\} \\
+ \sum_{i=1}^{m} x_i \times D_i
\]

Case Study: PNUTS

Distributed object/tuple store for Yahoo!
*Where in the World is My Data?* Kadambi et al., VLDB 2011

Optimization

Can add more complications
Multiple fragments
Fragment sizes
Concurrency control cost

Case Study: PNUTS

Issues
Where to locate data
What and where to replicate
Case Study: PNUTS

Policy constraints
MIN_COPIES: the minimum number of full replicas of the record that must exist

INCL_LIST: an inclusion list — the locations where a full replica of the record must exist

EXCL_LIST: an exclusion list — the locations where a full replica of the record cannot exist

Example rule 1
IF TABLE_NAME = 'Users'
THEN SET 'MIN_COPIES' = 2
CONSTRAINT_PRI = 0

Example rule 2
IF
TABLE_NAME = 'Users' AND
FIELD STR('Home Location') = 'France'
THEN
SET 'MIN_COPIES' = 3 AND
SET 'EXCL LIST' = 'USWest, USEast'
CONSTRAINT_PRI = 1

Summary
Description of fragmentation
Good fragmentations
Design of fragmentation
Allocation