UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

CS411 - SQL: Views and Indexes
Midterm review

illinois.edu

Announcements

- MP1 resubmission
  - if you do it, no extra credit points!
- Midterm 1 on Friday
- Ungraded SQL assignment available
  - prepare for exam
  - prepare for MP2
  - help for project

Views

- Subqueries can be used a lot
- ***Views*** are a way to query them like tables
- Two types
  - Virtual
    - do not exist physically
    - queried again each time
  - Materialized
    - physically stored on disk

Views

- Syntax

```
CREATE VIEW Villians AS
   SELECT * FROM JEDI
   WHERE side=“Dark”;
```

Views

- Useful for breaking down complex queries
- "Find the last names of all the people who have the same first name and the length of their last names differs by more than 1."
  1. Find the names of all the people
  2. Find the names of all the people with the same first name
  3. Find all the last names with different lengths

Updateable Views

- Some views can be updated
  - Require certain circumstances:
    1. No subqueries in WHERE clause
    2. Only one relation in FROM clause
    3. Any attributes outside of the view must allow NULL
- If we want to update other views, we should use an INSTEAD OF trigger

View Demo

Materialized Views

- If we use a view a lot, we can physically realize it on the disk
- Pros:
  - Faster to execute queries
- Cons:
  - More disk usage
  - Updates to underlying tables slower

Materialized Views

- Syntax

```
CREATE MATERIALIZED VIEW
Villians AS
   SELECT * FROM JEDI
   WHERE side=“Dark”;
```

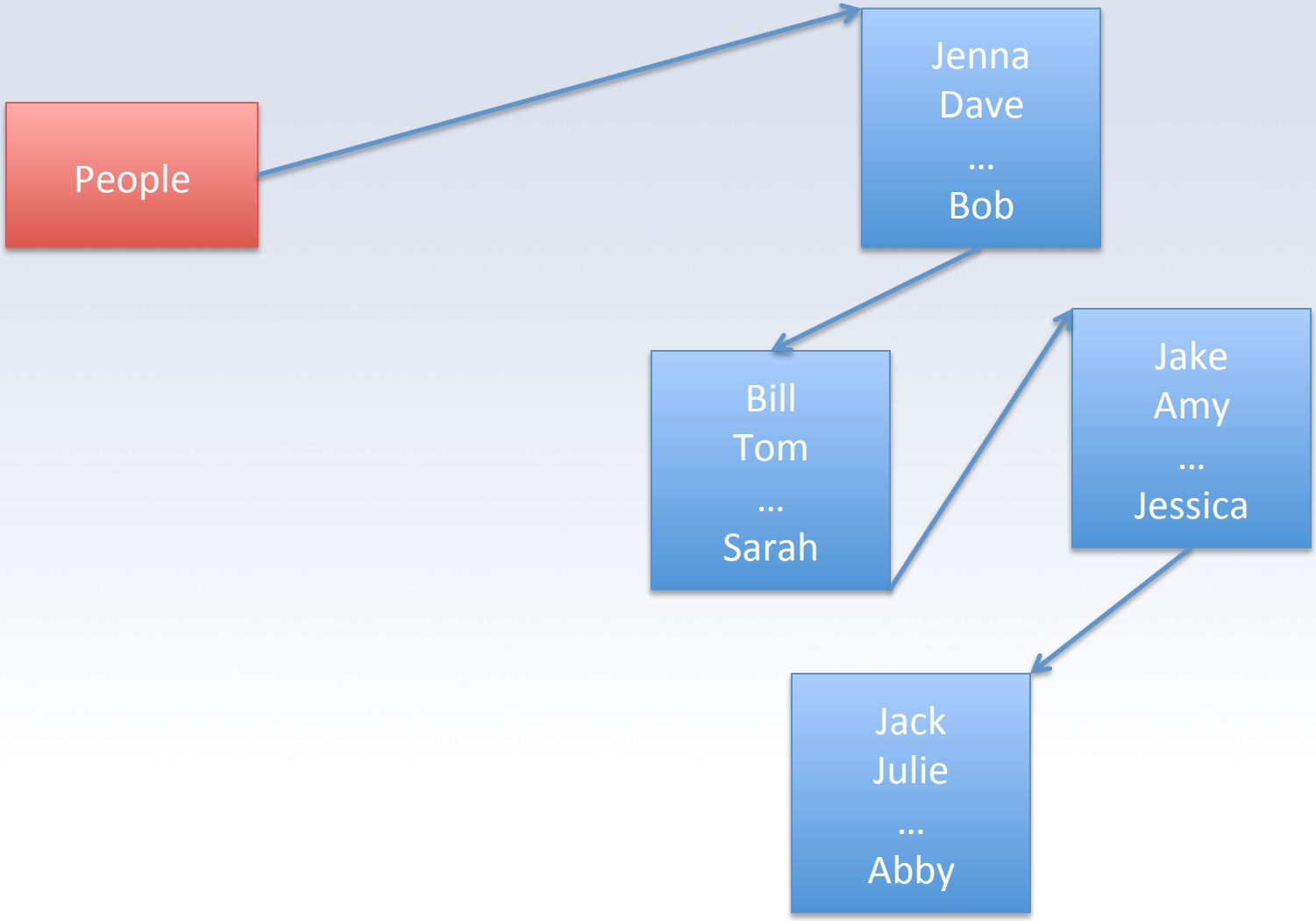
Materialized Views

- Used in enterprise scale DBMS implementations like Oracle and SQL Server
- Not supported in MySQL or sqlite

Indexes

- In reality, our tables are scattered across the disk
- Searching for specific tuples is inefficient

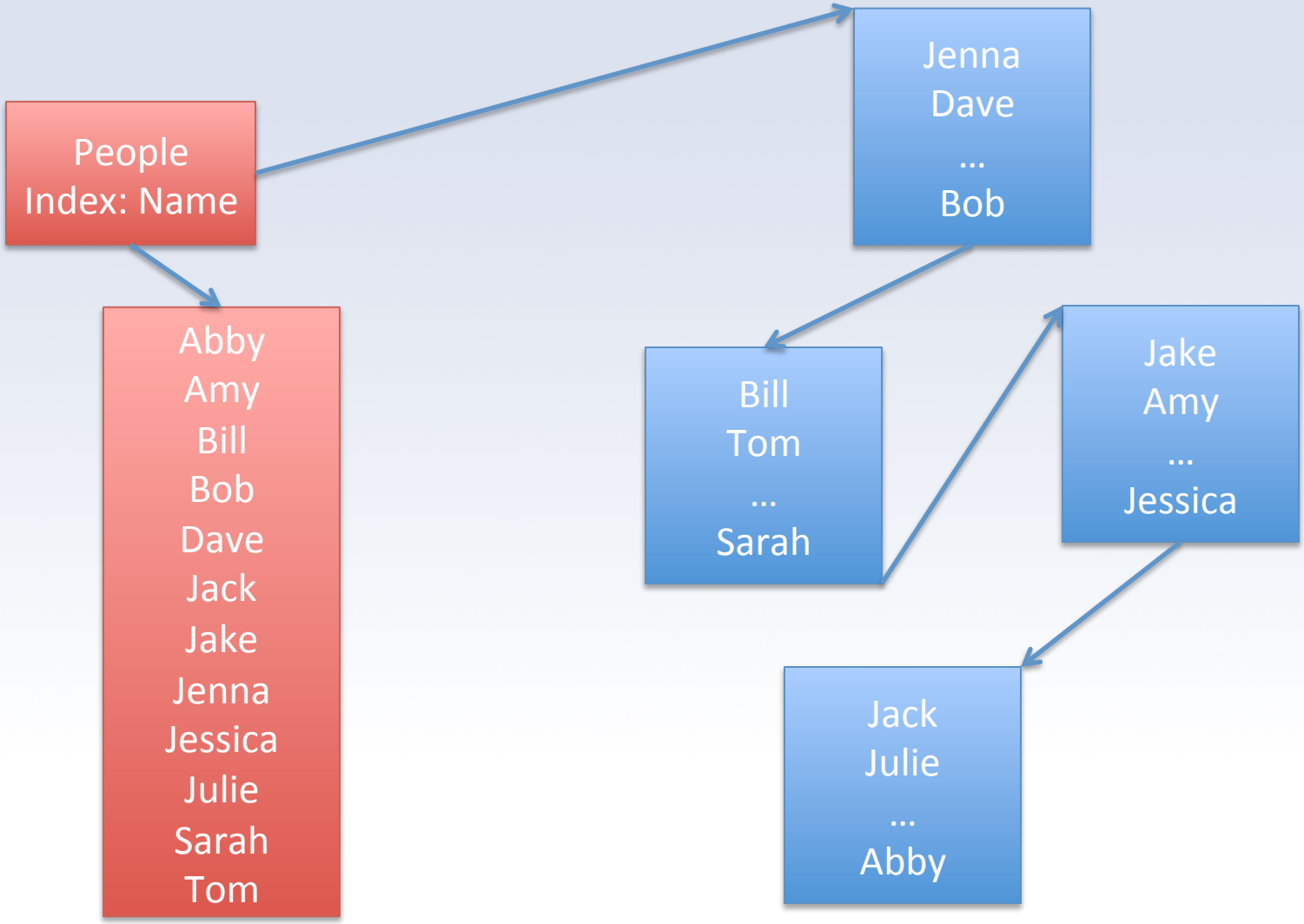
Indexes


People
Jenna
Dave
…
Bob
Bill
Tom
…
Sarah
Jake
Amy
…
Jessica
Jack
Julie
…
Abby

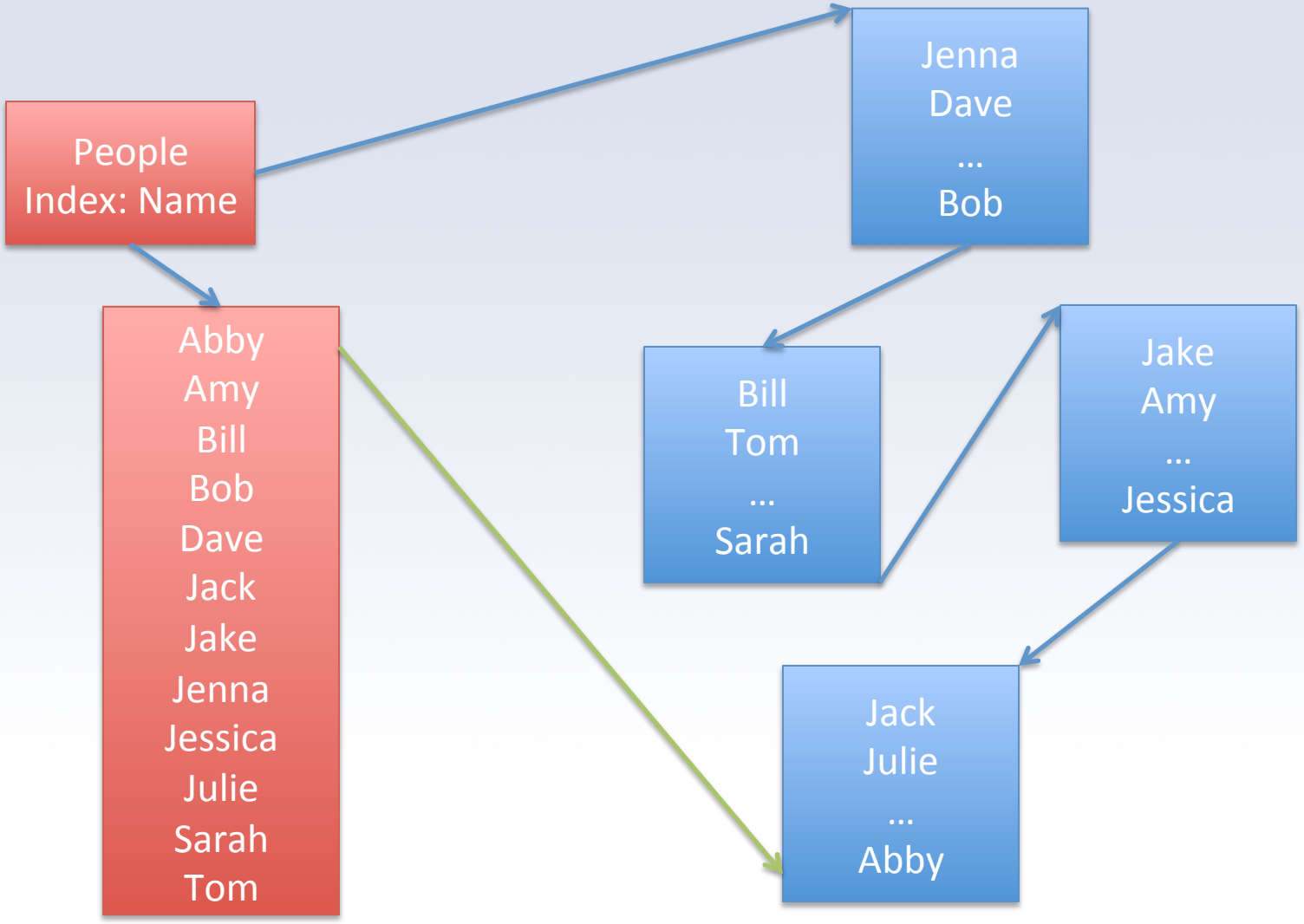
Indexes

- We can speed this process up using by ***indexing*** our table
- An ***index*** is a searchable structure that maps a series of attributes to actual tuples
- Queries using these attributes will execute faster

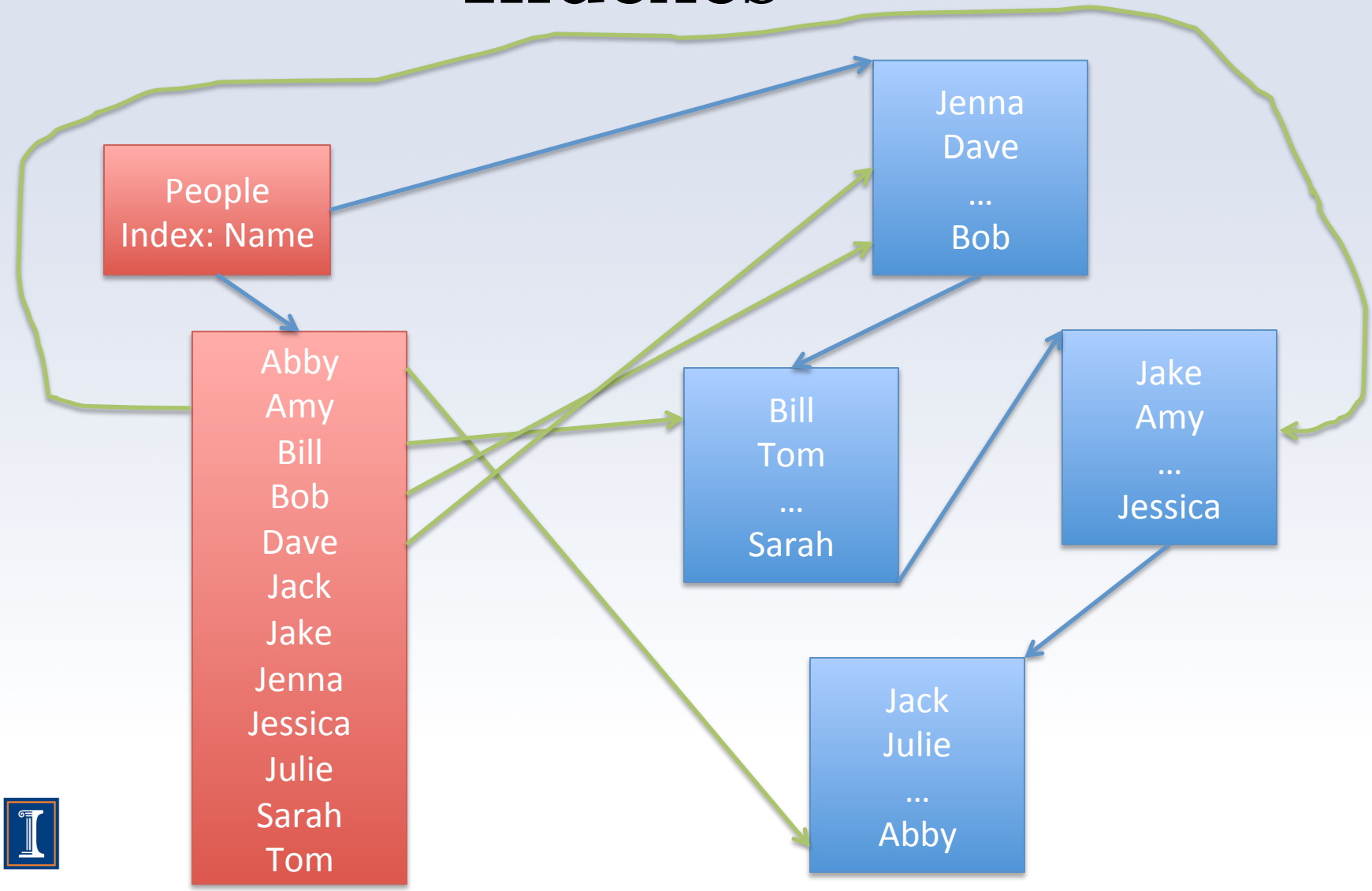
Indexes


People
Index: Name
Abby
Amy
Bill
Bob
Dave
Jack
Jake
Jenna
Jessica
Julie
Sarah
Tom
Jenna
Dave
…
Bob
Bill
Tom
…
Sarah
Jake
Amy
…
Jessica
Jack
Julie
…
Abby

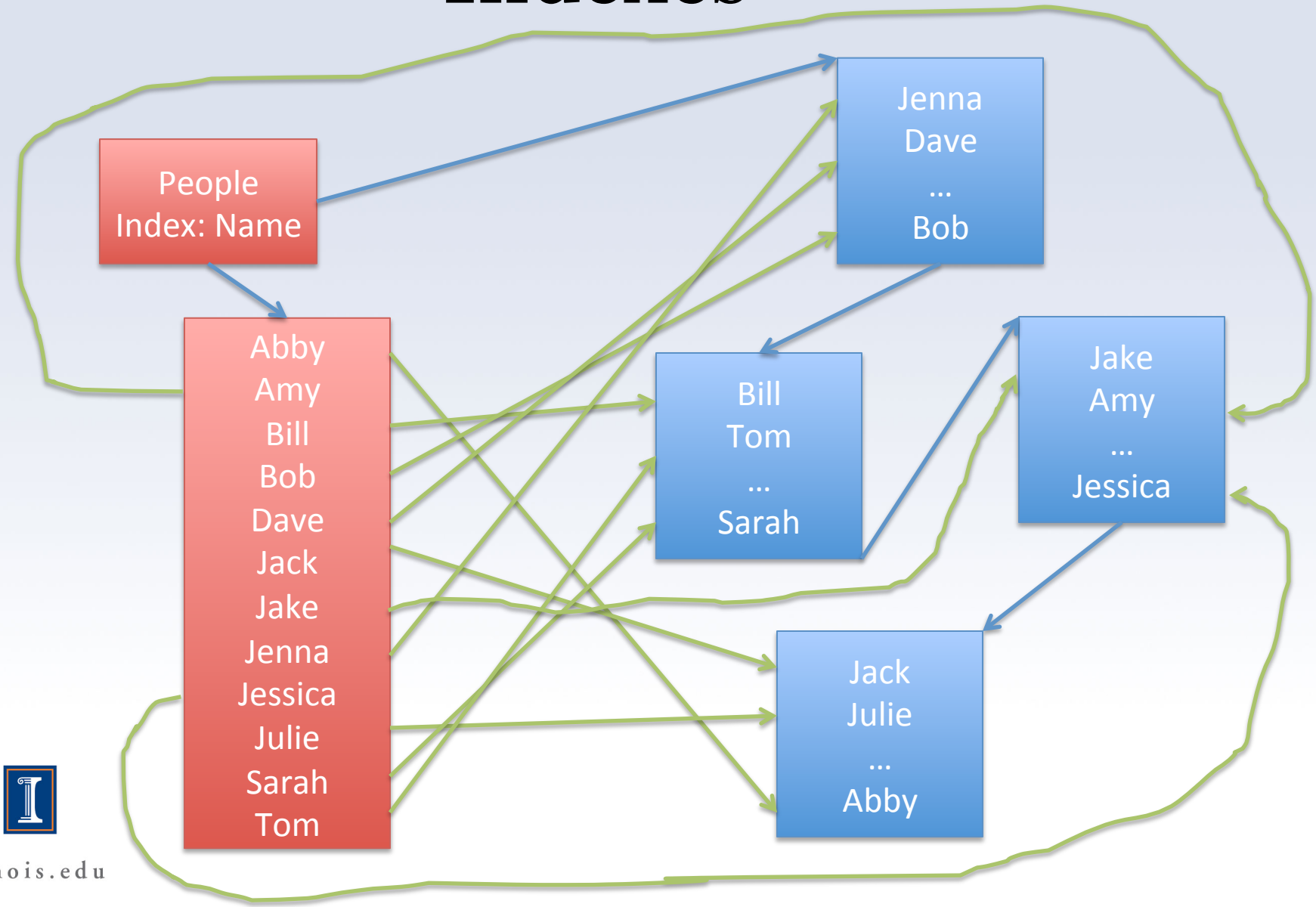
Indexes


People
Index: Name
Abby
Amy
Bill
Bob
Dave
Jack
Jake
Jenna
Jessica
Julie
Sarah
Tom
Jenna
Dave
…
Bob
Bill
Tom
…
Sarah
Jake
Amy
…
Jessica
Jack
Julie
…
Abby

Indexes


People
Index: Name
Abby
Amy
Bill
Bob
Dave
Jack
Jake
Jenna
Jessica
Julie
Sarah
Tom
Jenna
Dave
…
Bob
Bill
Tom
…
Sarah
Jake
Amy
…
Jessica
Jack
Julie
…
Abby

Indexes


People
Index: Name
Abby
Amy
Bill
Bob
Dave
Jack
Jake
Jenna
Jessica
Julie
Sarah
Tom
Jenna
Dave
…
Bob
Bill
Tom
…
Sarah
Jake
Amy
…
Jessica
Jack
Julie
…
Abby

Syntax

- We can create indexes in SQL as follows:

```
CREATE INDEX <IndexName>
ON <Table(attributes)>
```

Indexes

- Pros:
  - Speed up queries
- Cons:
  - Requires more disk space
  - Slows down modification
- We'll investigate this in depth later

Indexes

- Multiple attributes can form an index
- Values for index don't have to be unique
  - Attributes indexed don't need to form a key

Indexes

- When designing indexes, we should:
  - Index on keys or “nearly” keys
  - Index on attributes we query a lot
  - Put more commonly queried attributes first
- These are just heuristics.
- A deeper understanding will come when we study implementation.

Index Demo

Exam Overview

- 5 questions
- 2 multiple choice
  - worth more than 1/3 of the points
  - covering ***all*** of the topics

Practice Exams

- Posted on the website
- We've covered mostly the same material
  - Fall/Spring 2008 are very good
  - Fall 2009 and Fall 2010 have good multiple choice questions

Exam Topics

- Relational Model
  - Remember all of the definitions
- Relational algebra
  - Write queries/constraints from a specification
  - Understand relational algebra queries/ constraints from a query
  - Remember all the definitions

Example

c. [10 points] List the people who have made two or more bets on Illinois to lose. For the instance given above, your answer should be:

Winslett
Reed

Solution:

$\pi_{BETS_1.Who}(BETS_1 \bowtie_{(BETS_1.Who=BETS_2.Who) and (BETS_1.Game!=BETS_2.Game) and (BETS_1.Outcome='L') and (BETS_2.Outcome='L')} BETS_2)$

$BETS_1$ and $BETS_2$ are both of type $BETS$.

RA Hint

- Don’t be afraid to create temporary tables or variables along the way.
- Break the task down for complex queries.

Example

(b) List all the makers who sell **only one** type of products (either PC or printer, not both). (*6 points*)

Solution:

$\pi_{maker}\text{Product} - \pi_{\text{P1}.maker}(\text{P1} \bowtie_C \text{P2})$
where, P1 and P2 are both type of Product
$C = (\text{P1}.model = \text{P2}.model \text{ AND } \text{P1}.type <> \text{P2}.type)$

(c) List all the makers who sell **at least two** different models of PC (*6 points*)

Solution:

$\pi_{\text{P1}.maker}(\text{P1} \bowtie_C \text{P2})$
where, P1 and P2 are both type of Product
$C = (\text{P1}.model = \text{P2}.model \text{ AND } \text{P1}.type = \text{P2}.type \text{ AND } \text{P2}.type = \text{“pc”} \text{ AND } \text{P1}.model <> \text{P2}.model)$

Exam Topics

- Functional dependencies
  - Armstrong's axioms/FD manipulation
  - Attribute closure
  - Identify keys/superkeys

Example

Problem 3 (*16 points*) Relational Schema Design

Consider a relation R(A,B,C,D,E), with FDs $AB \rightarrow C$, $C \rightarrow A$, $C \rightarrow BD$, $D \rightarrow E$

(a) List all the keys of R. Do not list superkeys which are not (minimal) keys. (*6 points*)

(b) Is this relation in BCNF? If you answer is yes, explain why it is. If you answer is no, decompose the relation into BCNF, showing your decomposition steps. (*10 points*)

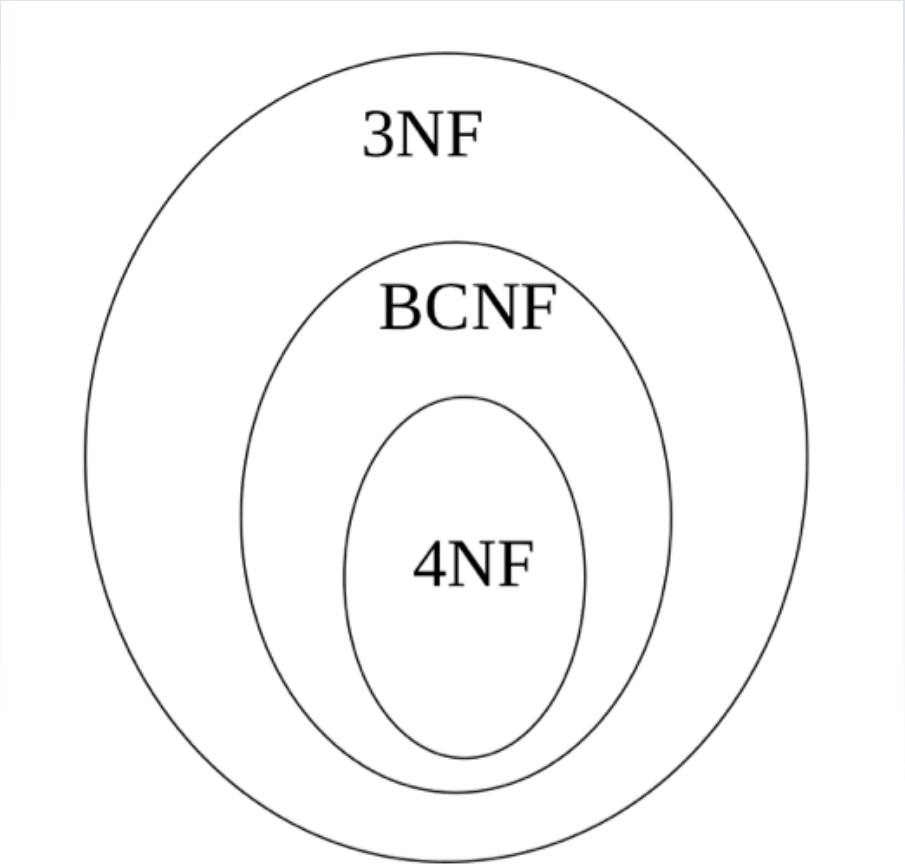
Exam Topics

- Normal Forms
  - BCNF decomposition
  - Hierarchy of normal forms
  - 3NF/4NF/MVD definitions
  - Properties of 3NF and BCNF

IMPORTANT

- All FDs are MVDs
- All relations in 4NF must be in BCNF
- Normal forms form a hierarchy

Normal Forms



Exam Topics

- E/R Modeling
  - Translate a description into an E/R diagram
  - Translate an E/R diagram to a relational schema
  - Understand all three of the ‘isa’ relation translation methods

Example

1. [20 points] Textbook bibliographical databases consist of information about publishers, books, authors, and citations, telling

1. which textbooks are published by which publisher;
2. which textbooks are written by which authors;
3. which textbooks cite which other text books.

Devise an entity-relationship diagram for this database. List the keys. Tell whether each relationship is many-many, many-one, one-many, or one-one. To keep your answer small, you only need to include one attribute per entity set in addition to its key attribute(s). You can assume textbooks come in one volume only, so you do not have to model volume information.

Exam Topics

- SQL
  - Queries, subqueries, aggregation
  - attribute/row constraints
  - keys and foreign keys
  - table creation/deletion/modification
- Triggers, indexes, assertions, and views
  - Don't need to write them
  - Understand what their purpose

SQL Hint

- Look at the answers for the Spring ‘08 SQL questions
- VIEWs can be very helpful for breaking down a task…

Example

3. [45 points]. Using the schema of problem 2, write the following queries in SQL. Your queries must return the correct answer for any instance of the database, not just the one given above.

a [10 points]. Who has bet on every game? For the instance given above, your answer must be empty relation.

Solution:

```
Create View AllGames As
(Select Game
from OUT)
UNION
(Select Game
from BETS)

Select Who
from BETS
Group By Who
Having count (distinct Game) =
(Select count (distinct Game)
form AllGames)
```

Example

c [20 points]. What game or games so far won the most money for people betting on Illinois to win? For the instance given above, your answer should be:

Michigan

: Solution:

```
Create View Success-Win As
Select Bets.Game, Sum(Bets.Amt) As SumAmt
From Bets, Out
Where Out.Game = Bets.Game and Bets.Outcome='W' and Out.Outcome='W'
Group By Game

Select distinct Game
From Success-Win
Where SumAmt >= ALL
(Select SumAmt
From Success-Win)
```

Exam topics

- Transactions
  - Relationship to ACID
  - Rollback vs. commit
  - Different isolation levels and read phenomena