notes

03 May 2012

Contents

1 DONE General

`CLOSED:` *2012-05-03 Thu 18:18*

1. Data Models: Notation for describing the structure of the data in the DB
2. Relational Model: Relations are tables representing info. Columns are headed by attributes. Rows are called tuples
3. Keys: A constraint on relations that uniquely identifies tuples in a table

2 TODO E/R (5)

2.1 Enitity-Relationship Model:

Description of entity sets, relationships amoung entity sets, and attributes of entity sets and relationships

2.2 Entity-Relationship Diagrams

We use rectangles, diamonds, and ovals to draw entity sets, relationships, and attributes, respectively

2.3 Multiplicity of Binary Relationships

2.3.1 One-to-One

Entity is one set is paired with an Entity from another set

2.3.2 Many-to-One

A single Entity from one set associates with many Entities from another set

2.3.3 Many-to-Many

Co-Many-to-One: Many to One both ways

2.3.4 Subclasses

The E/R model uses 'isa' to represent the fact that one entity set is a special case of another

- Similar with inheritance as Objects

2.3.5 TODO Stategies for E/R conversion

3 TODO Schema Design (10)

3.1 Functional Dependencies

If A_j, B_j are attributes of R a FD on R $\equiv A_1A_2...A_n \rightarrow B_1B_2...B_m$

3.2 Key Relation

$A = A_1, A_2, ...A_n$ is a key iff

1. If B is a set of all attributes in R then A is a key if $A \rightarrow B$
2. No subset of A, call A', satisfies $A' \rightarrow B$

3.3 Super Key:

a set B that contains the key A. This means it only satisfies condition 1 of the key relation

3.4 Closure

Set of all attributes impliable based on a givenset of attributes

3.5 TODO Minimal Basis

3.6 Boyce-Codd Normal Form (BCNF)

A relational schemem R, is in BCNF iff $\forall$ dependcies $X \to Y$, at least one of the following is true:

1. $X \to Y$ is a trivial FD ($X \subseteq Y$)
2. X is a superkey for R
    (a) Eliminates redundancy
    (b) May cause loss of FD's

3.7 Lossless-Join Decomposition:

A useful property of a decomposition is that the original relation can be recovered exactly by taking the natural joing of the relations in the decomposition

3.8 Dependency-Preserving Decomposition:

A decomposition that we can check for all the functional dependencies that hold in the original relation

3.9 Third Normal Form (3NF):

$X \to Y$ iff

1. $X \to Y$ is a trivial FD ($X \subseteq Y$)X $\subseteq Y$

3. X is a superkey

4. $A - X$ is a subset of a canidate key

- Preserves functional dependencies
- May not remove all redundancies

3.10 The Chase:

Algorithm for testing for a lossless-join

4 DONE Relational Algebra (5)

`CLOSED:` *2012-05-03 Thu 18:18*
This algebra underlies most query languages for the relational model.

4.1 Union ($\cup$)

Same as set theory

4.2 Intersection ($\cap$)

Same as set theory

4.3 Difference ($-$)

Same as set theory

4.4 Selection (σ)

1. Syntax: $\sigma_C(R)$, where $C =$ boolean condition on attributes.
2. Returns: Relation where all attributes satisfy C

4.5 Projection (π)

1. Syntax: $\pi_A(R)$, where $A =$ list of attributes
2. Returns: Relation with only columns corresponding with A

4.6 Caresian product ($\times$)

Same as set theory

4.7 Theta-Join ($\bowtie_\theta$)

1. Syntax: $R_1 \bowtie_\theta R_2$, where θ is an boolean attribute condition
2. Return: Relation with all joint tuples that satisfy θ

4.7.1 Natural Join ($\bowtie$)

1. $C =$ All common attributes columns are equal

4.8 Renaming (ρ)

4.9 DONE Bags

`CLOSED:` *2012-05-03 Thu 18:18*
Same as sets except duplicate elements are allowed

4.9.1 Relations as Bags

Most modern DB's implement relations as bags and not sets. This makes many common operations faster but pushes the burden of avoiding duplication on the designers

4.9.2 Extensions to relational Alg

To match the capabilities of SQL, some bag operations must extend the standard relational alg

- Duplicate-Elimination (δ)

  1. Turns the bag into a set

- Aggregation

  1. Summarize a column of a relation
  2. Typical aggregation ops: $SUM(A), AVG(A), COUNT(A), MIN(A), MAX(A)$

- Grouping (γ)

  1. Syntax: $\gamma_L(R)$

2. Partitions relation into groups based on a list of Attributes L

- OuterJoins ($\overset{o}{\bowtie}$)

1. Syntax: $R_1 \overset{o}{\bowtie}_\theta R_2$
2. Does a $\bowtie_\theta$ and then adds dangling tuples (tuples that have no corresponding slot in other relation padding attributes with NULL)

- Sort(τ)

1. Syntax: $\tau(R, L)$
2. Returns R sorted by list of attributes (Extra attributes are used for tie breaking)

5 TODO SQL (10)

The language SQL is the principal query language for relational database systems.

6 TODO SQL Constraints (5)

6.1 Comparison of constraints

Type of constraint	Where Declared	When Activated	Guaranteed to Hold?
Attribute-based CHECK	With Attribute	On inseration to relation or attribute Update	Not if subqueries
Tuple-based CHECK	Element of relation schema	On insertion to relation or tuple update	Not if subqueries
Assertion	Element of database schema	On any change to any mentioned relation	YES

6.2 Referential-Integrity Constraints;

1. Declaration that a value appearing in some attribute(s) of a set must appear in the corresponding attribute(s) of some other relation
2. Syntax: A REFERENCES R(A_2)
3. Syntax: FOREIGN KEY (<attributes>) REFERENCES <table>(<attributes>)o

6.3 Attribute-Based Check Constraints

1. Constraint on the value of an attribute by adding CHECK <condition> to be checked on the attribute

6.4 Tuple-Based Check Constraints

1. Containt on the tuples of a relation by adding CHECK <conditon> to be checked on the entire relation/tuple

6.5 Modifying Constraints:

1. A tuple-based check can be add/deleted using ALTER

6.6 Assertions

1. Delcaration of an assertion as an element in the database schema
2. The delcaration give a condition to be checked:
   (a) May involve multiple relations and may invole the relation as a whole

6.7 Triggers:

1. Event Driven (e.g. insertion, deletion, or update) check check of a condition ,which if true, starts a subroutine with any valid SQL statements

7 TODO Views

7.1 Virtual Views

1. Definition of how a virtual relation (aka view) can be constructed from other relations.
2. Once defined the can be treated as effective relations
   (a) Write operations only work in a restricted case, see Updatable Views

7.2 TODO Updatable Views

7.3 Instead-Of Triggers:

1. Because many views are not writable, SQL provides this trigger to switch how a tuple gets written into the database

   (a) Useful for emulating Table like functionallity in views

8 DONE Indexing (5)

`CLOSED:` *2012-05-03 Thu 18:18*

8.1 Clustered

1. All or most of the relation is sequentially set up on neightboring

parts of blocks

8.2 Dense Indexes

1. Index in which there is a key-pointer pair for every record in the data file (could be a tuple or more indexes)

8.3 Sparse Indexes

1. Index in which there is a key-pointer pair for every block in datafile

   (a) Only useful if the data is clustered and thus the position of the next element in the block can be inferred by the current position in the block

8.4 Multilevel Indexes

1. Indirection in the indexes (indexes point to indexes) which often saves space and allows for more complex datastructures

8.5 Inverted Indexes:

1. The relation between documents and the words they contain is often represented by an index structure with word-pointer pairs.
2. The pointer goes to a place in a "bucket" file where there is a list of pointers to places where the word occurs

9 TODO B+ Trees (5)

Like a B tree however no data is stored in

10 TODO Extensible hash tables (5)

11 TODO KD-Trees (5)

12 TODO Query Processing (15)

1. Method in which queries are compiled, which involves extensive optimization followed by Execution

12.1 Query Plans

1. Queres are compiled into <u>logical plans</u> often modeled after relation algebra, and then conververted into a <u>physical plan</u>
   (a) Logical Plan: Relational algebra like representation of query
   (b) Physical Plan: Specific algorithm to implement logical plan

12.2 Scanning

12.2.1 Table Scanning

1. Read each block holding tuples of the relation

12.2.2 Index Scanning

1. Utilize an index over an attribute to scan tuples in sorted order

12.3 Cost Measures for Physics Operators

12.3.1 Parameters for measuring Costs

	M	B(R)	T(R)	V(R,a)
Meaning:	# of Mem Blocks Avaliable	Blocks needed for R	# of tuples	# of unique values

12.3.2 Join Cost

$$T(A \bowtie B) = \frac{T(A)T(B)}{max(V(A,x), V(B,x))} \tag{1}$$

12.4 TODO One-Pass Algorithms

Operators	Approximate M required	Disk I/O
σ, π	1	B
γ, δ	B	B
$\cup, \cap, -, \times, \bowtie$	$min(B(R), B(S))$	$B(R) + B(S)$
$\bowtie$	$M \geq 2$	$\frac{B(R)B(S)}{M}$

1. If one argument fits in main mem, one can execute read the smaller relation to mem, and read the other argument a block at a time

12.4.1 TODO Nested-Loop Join

1. Simple join algorithm works even if neither arguments fit in main mem.
   (a) It reads as much as it can of the smaller relation into mem
   (b) Compares that with the entire other argument a block at a time

12.5 TODO Two-Pass Algorithms

Most algorithms for arguments that are to large to fit into mem are either sort-based hash-based or index-based

12.5.1 Sort-Based Algorithms

Operators	Approximate M required	Disk I/O
τ, γ, δ	$\sqrt{B}$	$3B$
$\cup, \cap, -$	$\sqrt{B(R) + B(S)}$	$3(B(R) + B(S))$
$\bowtie$	$\sqrt{B(R) + B(S)}$	$3(B(R) + B(S))$

1. Partiton argument(s) into main-mem-sized, sorted sublists
2. Sorted sublists are merges accordingly

- Two-Phase, Multiway Merge Sort (TPMMS)

- Diplicate Elimination

- Join

12.5.2 Hash-Based Algorithms

Operators	Approximate M required	Disk I/O
γ, δ	$\sqrt{B}$	$3B$
$\cup, \cap, -, \bowtie$	$\sqrt{B(S)}$	$3(B(R) + B(S))$
$\bowtie$	$\sqrt{B(S)}$	$(3 - \frac{2M}{B(S)})(B(R) + B(S))$

1. Use a hash function to partition arguments into buckets.
2. Apply operation to each bucket individually (unary) or in pairs(binary)

12.5.3 Hashing vs Sorting

1. Hashes are often faster since they require only one arg to be small
2. Sort-Based are convenient if data needs to be sorted either while working or outputing

12.5.4 Index-Based Algorithms

1. Speed up selections when applicable
2. If one relation is 'small' and the other has an index on join attribute then index-based algorithms are often quite fast