CS 338 course note

Chenxuan Wei

Sep 2022

Contents

1 Introduction to database

1. Terms
   Data redundancy: presence of duplicate data in multiple data files
   Data inconsistency: the same attribute may have different values

2. Database
   a collection of related information stored in a stuctured form

3. DBMS:
   a collection of programs that manipulate a database

4. Data Model

   - Relational Model
   - Object-oriented model
   - semi-structed data model
   - network model
   - Hierarchical model

5. Schema

   - Physical schema: database at physical level
   - logical schema: database at logical schema
   - External schema: database at external schema

2 Relational

1. Terms

   - attribute: each column with in a table
   - domain: all possible value of a attribute
   - Primiary key: a attribute in a row that must be unique in a table
   - Tuple: rows
   - Schema of a relation: definiton of a table
   - a instance: table content

2. Integrity Constaints
   is a condition that must be true for any instance of the database
   Domain constrain: must satisifeid domain
   Primary key constraints: each relation must have a primary key, and they must be unique
   Foreign key: set of filed in one relation used to refert to a tuple in another relation

3 Relational algebra and calculus

1. Relational Quesry language
   A major strengh of the relational model: supports simple, powerful querying of data

2. Relational algebra
   Result of a retrieval is new relation
   sqquence of relational algebra operations forms a relational algebra expression

3. Operations

   - selection (σ): select a subset of rows from relation
   - projection(π) deletes unwated columns from relation
   - cross-product(X) allows us combines 2 relation
   - Set-difference (-) tuples in relation1 but not 2
   - Union(Y) tuple in one of 1 or 2

   Format: (operation)$_{boolean}$ (relation)

4. Boolean
   used to show true value

5. Assignment operation
   $< -$ allowed to assign variable

6. Union compatible
   if 2 relation have the same degree and all attributes are defined on same domains

7. Foreign key
   Assume R1(ABC), R2(EFG) there is a FK: R1.A referrece R2.G
   the value of R1.A must be
   Null or unique in R2
   however, R2.G does not need to be PK

8. Rename operation (useless)
   format: $p_{(relation)}(relation)$ or $p_{(col,col)}(relation)$
   the first one rename relation, but the second one only rename column

9. Join operation
   symbol: $\bowtie$
   a combination of cross product and selection, notice must have different attributes name
   The following are the same:

   - $e < -R1XR2$
     result $< -\sigma_{bool}(e)$

- R1 (join)$_{bool}$(R2)

10. Natural join operation
    result $< -R1 * R2$
    Assume $R(ABC), S(AD)$, $R * S- > (ABCD)$
    will auto=same attributes, and combine attributes, also allowed same attribute name

11. Division Operation
    Assume $R1(r1_i)$, $R2(r2_i)$, $R1 \div R2 =$
    $(r1_i)$ such that $r1_i \notin R2$ and keep all tuple that all not included $r1_i$ appear in R2

12. Aggreation:
    ${}_{G_i}g_{f_i(A_i)}(E)$, allowed optional As to change the name of function F1
    function includes

    - avg
    - min
    - max
    - sum
    - count

4 SQL mannipulation

4.1 Data mainipulation

1. select basic format
   select (attribute) **from** (table) **where** (condition)
   if mutiple table selected, they will be cross prodected
   can use table**.**attribute to for duplicate column namess
   where, order by, group by,having must be in this order

2. rename
   can rename attribute name **AS**
   can give table temp name right after it's name

3. **distinct**
   a key word to eliminate duplicates in rows
   usage: **select distinct** (attributes)......

4. nested query
   when nest a table in from, must give the table a name
   when used in where, no need to give name

5. **join**
   usage: (table) **join** (table) **on** (condition (only equality))

6. **natural join**
   usage: (table) **natural join** (table)
   other join is the same by different name

7. **Like**
   compare text value in pattern
   % compare zero or more characters
   _ compare exactly one character

8. **IN** and **NOT IN**
   check if the attribute value is in the subsequence table

9. explicit sets
   like (1,2,3) for in and not in

10. **exsits**
    will return true if the table have atleast one row

11. **Unique/not unique**
    not supported in SQLite
    will check if there is any duplicate rows

12. **any** and **all**
    used with compare operation like (<)

13. **order by**
    sort result on one or more of attribute
    from small to big
    used desc to reverse

14. **group by**
    include grouping attributes
    if used, **select** (attribute) can only include aggregation function and groupting attributes

15. **having**
    is like use aggregation in where

16. **union** and **intersection**, minus
    (q1) union/intersect/except (q2),

4.2 Data modification

1. Create table
   Create table table name (Attribute Domain, or integrity-constraint)

2. Domain type

   - char(n): a fixed length string
   - varchar(n): not fixed string length with maximum length n
   - int: integer
   - smallint: small integer
   - numeric(p, d): fixed point number: p is digit, n is the position of decimal
   - read, double precision: floating point and double precision floating pointnumbers
   - float(n): floating point number with n is digit
   - not null: can't be null
   - customed domain: create a specificy domain

3. Date/time type

   - date: date
   - time: Time with day, hour minutes and second
   - timestamp: date+time
   - Interval: period of time

4. Integrity Constraint in SQL I

   - not null
   - primary key (A_i)
   - check (P): p is a condition

5. Foreign key
   Foreign key (A_i) **References** R(b_i)
   allow A_i refer to $R(B_I)$

6. Drop table
   Drop table simply remove the table from databse with all information

7. Alter table
   is used to add/change attibute type,domain
   Alter table (r) add (A D) drop (A D)

8. Delete
   Deleta from R **where** P
   delete row from R where satisified P

9. Insert
   Insert into R **values** (v)
   v must match the correct order of R's attributes

10. Update
    Update R **set** (attribute = expression) **where** (condition)

11. Case
    Case when then else end

4.3 advance topic

1. Views
   create a ”temp” table
   create view view name **as** query

2. Why views
   view help to create data

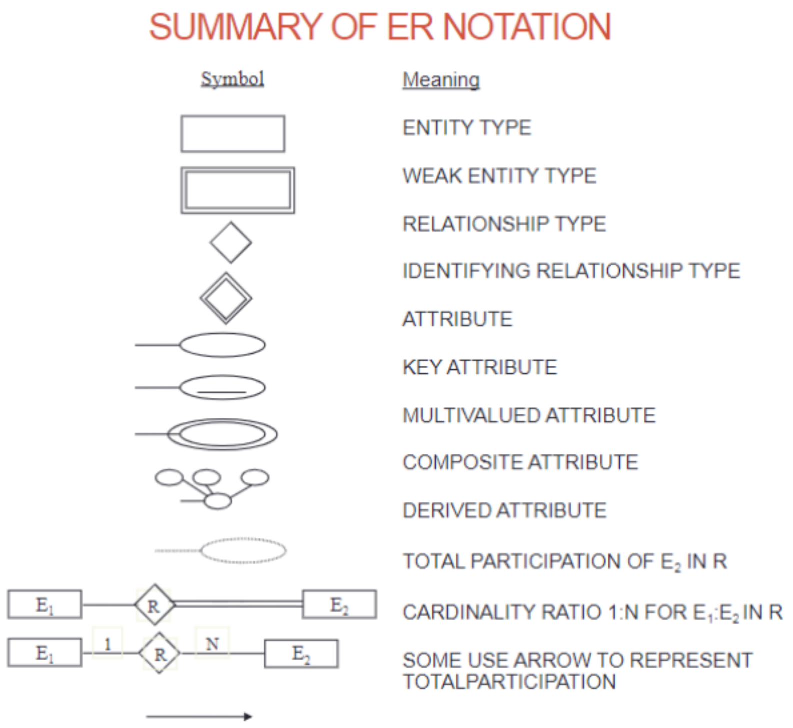
3. Assertion
   create assertion (name) **check** (condition)

4. Triggers
   create trigger (name) **after** (some condition) (event)

5 ER

5.1 Basic

1. Entity (square)
   Real-word object distringuishable from other obejcts
2. Entity Set
   A collection of similar entites
3. Attribute (oval)
   a entity represent a set of attributes
4. Type of attributes
   - Simple: one atomic value
   - Composite: a attribute composed of several components
   - Multi-valued: an entry may have multiple values for the attribute
5. Keys
   - Super key
     an entity set is a set of one or more attributes whose values uniquely determine each entity
   - Canadidate key
     of an entity set is a minimal super key
   - primary key (underline)
     is when canadidate key have only one attribute
6. Relationship (ling xing)
   connected between 2 entity with a name and some attributes
7. Cardinality
   - 1-1
     means that a enity can only be connected with only one other entity
   - 1-many
     means that the a object can be associate with many other entity
   - Many-many
     mneas that many can associate with many entity
8. Participation Constraint
   - Total participation (double line connected to the diamind)
     every entity in the entity set participate in at least one other entity
   - Partical participation
     can have no relation
9. Week entity (double rectangle)
   Does not have a primary key
   must be total participate within a relationship



SUMMARY OF ER NOTATION

Symbol

Meaning

ENTITY TYPE

WEAK ENTITY TYPE

RELATIONSHIP TYPE

IDENTIFYING RELATIONSHIP TYPE

ATTRIBUTE

KEY ATTRIBUTE

MULTIVALUED ATTRIBUTE

COMPOSITE ATTRIBUTE

DERIVED ATTRIBUTE

TOTAL PARTICIPATION OF E_2 IN R

CARDINALITY RATIO 1:N FOR E_1:E_2 IN R

SOME USE ARROW TO REPRESENT
TOTALPARTICIPATION

5.2 Mapping

1. Basic Principles

   - No loss of information
   - Minimal redundancy
   - Minimize the use of NULL

2. Mapping steps

   - Step1: Mapping of regular entity types
   - Step2: Mapping of MUltivalued attributes
   - Step3: Mapping of Week Enity Types
   - Step4: Map 1:1 relationship
   - Step5: Map 1:N relationship
   - Step6: map M:N relationship
   - Step7: Map N-ary relationship types

3. Step1:
   For each strong entity, create a relation R, and include all simple attribute
   break composite attribute
   PK are still PK

4. Step2:
   For each multivalued attribute A belong to S, create a new relationship C such that C have 2 cloumn, one for A of for FK to PK of S
   PK for C is A+FK

5. Step3: weak entity
   For each weak eneity W , create a relation R, include the PK of owner entity E
   PK of R is: FK from owner + partial key of W

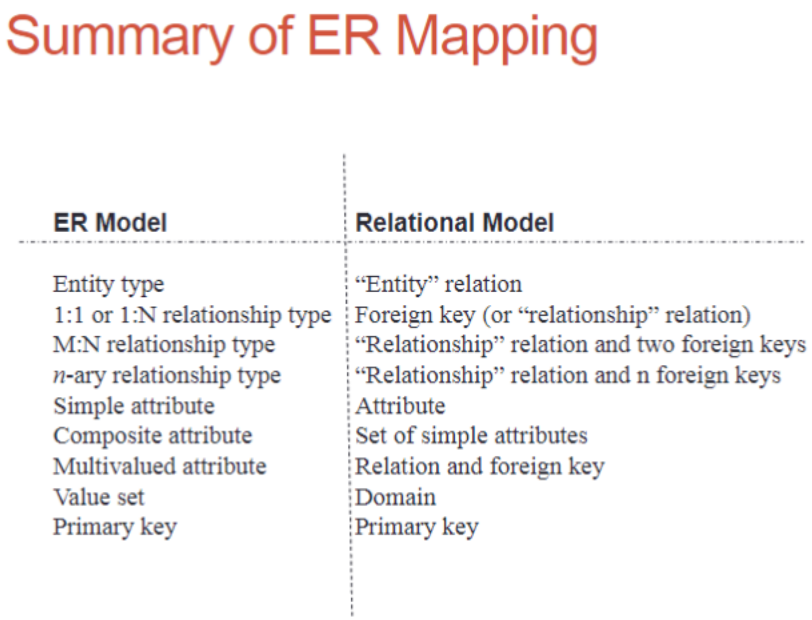
6. Step4
   for each 1 to 1, have 3 way:

   - Both total: combine both relation to 1 attribute but only remain one of the PK
   - One total: add a FK of PK from the 1 side to N relation
   - No total: create a new relation

7. Step5:
   in the N side, include a FK from the 1 entity

8. Step6
   Create a new relation include PK from both entity
   the PK in new relation is the combine of both PK from entity

9. Step7
   For each n-ary relationship type R, create new relation C to represent R
   include all PK from all participant, combinition is the PK of C



Summary of ER Mapping

ER Model	Relational Model
Entity type	“Entity” relation
1:1 or 1:N relationship type	Foreign key (or “relationship” relation)
M:N relationship type	“Relationship” relation and two foreign keys
n-ary relationship type	“Relationship” relation and n foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Primary key	Primary key

6 Normalization

1. Why needed
   Redundancy

2. Functional Dependency
   Basis for normlization
   $A- > B$
   where A, B are attributes
   where A's value can determine b's value
   A B must be different attributes

3. Good design
   left hand side of FD is always CK

4. Goal of Decomposition

   - Lossless
     do not loss any information

   - Dependency preservation
     all of the non-trival FDs each end up in just one relation

   - Boyce-Codd normal form
     no redundancy beyond goreign keys

5. Normal forms
   Where the relation satisifiy a certain condition

6. Different levels

   - 1NF
     all attribute values are atomic (part of definition of relational model

   - 2NF
     all non-key attributes must depend on a whole candidate key
     (No partial dependencies)

   - 3NF
     Table is in 2NF and all non-key attribtues must depend on only a canadidate key(no transitivei depdneces)

   - BCNF
     every determinant is a super key

   - BCNF > 3NF > 2NF > 1NF

7. Compute attribute closure
   given a starting key S = (a)
   add any functional dependency $a- > b$ to the set S
   recursily add function depdency of b to S
   until there is no more to add

8. BCNF decomposition
   fist find a key that it's attribute closure is not the entire relation
   decomposit to

   - $R1->$ that attribute clusure
   - $R2->$ rest of attributes + the key

7 Transaction

1. Transaction
   is a unit of program execution that accesses and possibily updates various data items

2. ACID properties

   - Atomicity
     Either all operations of the transaction are properly reflected in the database or none are
   - Consistency
     Database constraints are preserved
   - Isolation
     Although multiple transactions may execute concurrently, each transaction must be unaware of other concurrently executing transactions
     itermediate transaction relsultes must be hidden from other concurrently exected transactions
     means that every transaction must happened one by one
   - Durability
     after a transaction aompletes, the changes it has made the database persist, even with system failures

3. Transaction state

   - Active - the initial state; transaction stats in this state while it is executing
   - Partially committed
     after final satatement has been executed
   - Failed
     after the discovery that normal execution can no longer proceed
   - Aborted
     AFter the transaction has been rolled back and the database restored to its state prior to the start of the transaction.. Tow option after it has been aborted
     restart or kill
   - Comitted
     after successful completion

4. commit
   cause a transaction to complete

5. Rollback
   cause the transaction to end, by aborting
   will have no effect to database (clear all previous not commited operation)

6. isolation levels
   sql have 4 isolation levels about choice about what interactionare allowed by transactions that execute at about the same time

7. Serializeable transaction (default)
   one set of statement must be done before another could run

8. Read-committed transactions
   can only see committed data, but not necessarily the same data at each statement

9. Reapeatbale-read transactions
   for every read, all previous readed data will be read again

10. Read uncommitted (dirty read)
    can read even transaction is not commited

8 Storage and indexing

8.1 better way to organize data

1. DBMS structure

   - Query optimization
   - Relational operator algs
   - Files and access methods
   - BUffer management (TLB)
   - Disk space management

2. Hash files (FYI)
   The files is divied into M equal-sized buckets, numbered from 0 to M-1
   a hash function h determines item K is stored in bucket i, where $i = h(K)$
   the hash function is very efficient, and insert K is also very efficient
   Collisions occur when a new record hashes to a bucket is full
   an overflow file is kept for sotring such records

3. B^+ tree file organization
   data file degradation problem is solved by using B^+tree file organization

4. indexes as Access paths
   A single-level index is an auxiliary file that makes it more efficient to search for a record in the data file
   index ususally specified on one attribute of the file
   more attributes is allowed
   create a (search key, pointer to record) in the file, which ordered by search key

8.2 Better way to store data

1. Costs of Accessing Disk
   the time of reading/writing a disk refer to input/output cost

   - Seeking time: time to move the arm to proper track
     4-10 ms, specified by manufacturer
   - ROtation time: time to rotate the disk to put the right sector under the read/write head
     depends on RPM, usually 4ms
   - Transfer time: time between memory and disk
     defined by interface, usually negligible

2. Optimize Disk Access
   Data is trenasferred between disk and main memory in blocks
   block is a contiguous sequence of sectors from a single track, size from 512 B - kB
   can minimize IO costs (like caching)

3. Different Level of RAID(redundant Arrays of Inexpensive Disks)

   - 0: no redundant data, striping only
   - 1: mirrored disks (prevent disk fail), mirror only, no striping
   - 5: block-level data striping, parity information across all disks, both

4. Data striping
   Data striping intentionally distributes data blocks of the same file over multiple disks to speed up access performance

5. Parity
   A partiy bit or check bit is a bitt added toa string of birnary code that indicates whether the number of 1-bits in the string (including the parity bit) is even or odd

6. RAID 5 parity method
   Assume we have n data block with k bit each block
   create a new block that it's ith position contain the even/odd parity bit of all ith position in other block
   so if one disk failed, we are able to recover it by some computation

7. Data backup

   - online backup
     instant real-time backup
     RAID 1 and 5
     protect against one HD fail

   - Offline backup
     donw at each day, copy and ship to different location
     protect against complete failure, but can only recover data from one day ago

8. ALternative Mass storage device
   SSD, Tape drive

9. HHD bs SSD

   - Fail rate
     HHD: 1.5 million hours
     SSD: 2.0 million hours
   - IO speed
     HHD: 50-120MB/s
     SSD: > 200 MB/s
   - File open speed
     SSD 30% faster
   - Affect by mangintsm
     HDD will be effected, SSD does not
   - Ruggedness
     HDD might be damaged by movement, SSD does not
   - Batter life
     SSD: 2-3 watts, HDD: 5-7 watts
   - Cost
     HDD: $0.03/gB
     SSD: $0.2-0.3/gB
   - Capacity
     HDD: around 500GB and 2TB, 10TB for desktops
     SSD: 1TB for notebook, 4TB for desktop
   - OS boot time
     HDD: 30-40s
     SSD: 8-13s
   - Noise
     HDD: loud
     SSD: no sound
   - VIbration
     HDD: yes, SSD: no

10. Tape drive

    - Pros
      3*chepaer than HDD
      require no electrical power, when not used

Faster than HDD
less sensitive to damage
30 year life
Can be not active for long time

- Cons
  High entry cost: need IO machine: over $1000
  not for random access
  need proper storage condition
  can be re-used for 100x at most
  require special driver