* Operators in RA
  + Set Difference (-)
    - Binary Operator
    - r&s -> relations
    - r - s
      * Tuples in r, that are not in s
  + Set Union(U)
    - Binary Operator
    - r&s -> relations
    - r U s
      * All tuples in both r&s that are unique and should have common attributes
  + Set Intersection (**∩)**
  + Natural Join (⋈)
    - Case 1
    - Case 2
  + 𝛳-join
  + ~~Division or quotient~~
* RA in SQL queries
* Examples
  + Book(AccNo, YrPub, Title)
  + Borrow(AccNo, CdNo, DOI)
  + User(CdNo, BName, BAdd)
  + Supplier(SName, SAdd)
  + Supply(AccNo, CdNo, DOI)
  + R-S
    - 𝛱AccNo(Book) - 𝛱AccNo(Borrow)
      * Find all titles of books not borrowed
  + RUS
    - 𝛱AccNo(supply) U 𝛱AccNo(borrow)
* 01/28/18
* Normalization
  + Process to eliminate redundancy as redundancy is the root cause of a number of problems
  + Anomalies
    - Insertion anomaly
    - Deletion anomaly
    - Updation anomaly
  + Normalization (Breakdown)
    - Functional Dependencies:
      * 1 NF
      * 2 NF
      * 3 NF
      * BCNF
* Functional Dependencies (FD)
  + Tool -> searches
  + Fd:
    - implies
  + Types
    - Trivial:
      * B is subset of a
    - Non-Trivial:
      * B is not a subset of a
    - Completely Non-Trivial:
      * A or B is absolute disjoint
  + Definition of Functional Dependencies: Given relation R, and 2 tuples, t1 and t2,
    - if t1[]=t2[] then
    - t1[]=t2[] holds
  + If all is unique, qualifies for primary key(if touches each value)
  + Definition of Closure of a Set of Attributes: Given an attribute “” what are the related attributes to given the FD’s
  + Hidden Types of FD
    - A ->b
    - b->c
    - Therefore a ->c : hidden FD
* 1/31/18
* Armstrong Axioms to determine closure of attributes
  + Input: Set of FD’s and the schema R(A, B, C, D
  + Rules
    - 1 Reflexivity Rule : if B , then A -> B
    - 2 Augmentation Rule: if A -> B then AC -> BC
    - 3 Transitivity Rule: if A -> B and B -> C, then A -> C
    - 4 Union Rule: if A -> B and A -> X, then A -> BX
    - 5 Decomposition: if A -> BX, then A -> B and A -> X
    - 6 Pseudo Transitivity: if A -> B and CB ->D, then CA -> D
  + Objective: Apply any of these rules to the FD’s until no new FD’s are obtained
    - Checks:
      * 1 Soundness -> Any new FD derived by the axioms is indeed a member of the closure
      * 2 Completeness -> all elements of the closure can be determined by repeated application of axioms
  + Example 1:
    - Given R(A, B, C)
    - F = {A -> B, B -> C}
    - Determine the closure of A, B, C
    - Solution
      * A+ =[ABC]
      * B+ =[BC]
      * C+ =[C]
  + Example 2:
    - Given R(A, B, C, D, E, F, G)
    - F={A->B,BC->DE, AEG->G}
    - Determine (AC)+
    - Solution:
      * (AC)+ = [ACBDE]
  + Example 3:
    - Given R(A, B, C, D, E)
    - *F* = {A->BC, CD->E, B->D, E->A}
    - Solution:
      * B+ = [B D]
  + Example 4:
    - Given R(A, B, C, D, E, F, G, H)
    - *F*={A->BC, CD->E, E->C, D->AEH, ABH->BD, DH->BC
    - Does BCD->H
    - Solution:
      * BCD+ =[BCDAEH]
      * Valid
* cover/Equivalence of FD’s
  + Let *F* and *G* be 2 sets of FD’s on a single relation R
  + Then *F* is a cover of G if *F*+ = *G*+
  + Example
    - R(A, C, D, E, H)
    - *F* = {A->C, AC->D, E->AD, E->H
      * Use rules of G
      * A+ = [ACD]
      * AC+ =[ACD]
      * E+ =[EAHCD]
    - *G* = {A->CD, E->AH}
      * Use rules of F
      * A+ =[ACD]
      * E+ =[EADHC]
    - Solution:
      * Since A+ and E+ is the same in both, the are equivalent
      * And extras(AC) don’t matter
      * Formal
      * A+F=A+G and E+F=E+G therefore F+=G+
* Finding candidate keys given FD’s
  + Edge Diagrams
    - R(A, B, C, D, E, F, G, H)
    - *F*={AB->C, A->DE, B->F, F->GH}
    - AB+=[ABCDEFGH]
      * Therefore AB is a candidate key
    - Draw lines from each first term to the second terms
      * If it has a lint to it it is part of set
      * If it has a line to it it is not part of solution, u
    - ???Candidate key the the smallest subset of keys that give all of them???
  + How-To
    - Draw lines according to *F*, use ones with arrows going out, not in
    - If that doesn’t cover all options, use ones with arrows both in and out
    - If need set of possible keys, just all combos with minimum(stop at 3)
  + Example
    - R(A, B, C, D, E, F, G, H)
    - *F*={AB->C, BD->EF, AD->G, A->H}
    - Solution = ABD
  + Example 2
    - R(A, B, C, D, E)
    - *F*={BC->ADE, D->B}
    - Solution BC or CD
  + Example 3
    - R(A, B, C, D, E)
    - *F*={AB->CD,D->A, BC->DE}
    - AB or BC
* 2/5/18
* Joins
  + Natural Join (Equi-join) (equal operator ->automatic)
  + ⁠Theta join (-> predicate for joining)
* Outer Joins
  + Extension of
  + Objective is to retain more information after joining
* Left Join
  + Retains every tuple from the relation on the left and obeys the
* 2/9/18
* Normal Forms
  + 3 Normal forms
    - 1st NF
      * Address multivalued attributes
        + MvA: (Ex) All courses a student is taking (many entries for single person, 1st NF breaks it into individual entries(multiple instances of each student)
      * Steps
        + Identify a primary key
        + Establish all FD’s with the primary key (basically make sure all attributes can be accessed from PK)

Partial FD: (can split each attribute to be IDd by one part of the primary key) (emp # gives name, job, pay…)

AB->all

A-> subset

B-> the rest

Transitive FD:

Partial inside of a partial

Or when you can get to 1 and it leads to another

* + - 2nd NF
      * Remove all Partial FD
        + Take each Partial Dependency and make it a table
    - 3rd NF
      * Remove Transitive FD
        + Make it another table
    - Boyce Codd NF
* 2/16/18
* Query Processing
  + Efficient query Processing is crucial for good or effective operations of a database
  + Depends on many factors
    - Type of query
    - Hardware,....
* Iterators