INTRODUCTION

A quick disclaimer before reading the notes: they were taken by me during the lectures, they do not replace the professor's work and are not sufficient for passing the exams. Moreover they might contain mistakes, so please double check all that you read. The notes are freely readable and can be shared (always remembering to credit me and to not obscure this page), but **can't** be modified. Thank you and hope these notes are useful!

-Francesca Cinelli

DMA V1

Proofs and Definitions for oral exam

For all the proofs please check the file with the class notes

UNION RULE (I)

- if $X \to Y \in F^A$ and $X \to Z \in F^A$ then $X \to YZ \in F^A$

proof:

- $X \to Y \in F^A$ by augmentation $X \to XY \in F^A$
- $X \to Z \in F^A$ by augmentation $XY \to YZ \in F^A$

} by transitivity $X \to YZ \in F^A$

DECOMPOSITION RULE (I)

- if $X \to Y \in F^A$ and $Z \subseteq Y$ then $X \to Z \in F^A$

proof:

- $Z \subseteq Y$ by reflexivity $Y \to Z \in F^A$
- $X \to Y \in F^A$

} by transitivity $X \to Z \in F^A$

PSEUDOTRANSITIVITY RULE (I)

- $X \to Y \in F^A$ and $WY \to Z \in F^A$ then $WX \to Z \in F^A$

proof:

- $X \to Y \in F^A$ by augmentation $WX \to WY \in F^A$
- $WY \to Z \in F^A$

} by transitivity $WX \to Z \in F^A$

Decomposition algorithm (II)

Theorem: the decomposition algo computes a decomposition ρ of R s.t. each relational schema in ρ is in 3NF and ρ preserves F

Proof:

→ ρ preserves F

let $G = \bigcup_{i=1}^{k} \pi_{R_i}(F)$; since for each f.d. $X \to A \in F$ we have that $XA \in \rho$, we have that this dependency of F will be in G, hence $F \subseteq G$ and $F^+ \subseteq G^+$

→ each schema is in 3NF

1. if $S \in \rho$ ($S = S \cup A$, A is not involved in any f.d. in F)
   each attribute in S is part of the key and so S is in 3NF

2. if $R \in \rho$
   there is a f.d. in F involving all attributes of R; as F is a minimal cover, this d. will have the form $(R-A) \to A$; as F is a minimal cover, it cannot exist a f.d. $X \to A$ in F such that $X \subseteq R-A$ and then $R-A$ is a key of R; let $Y \to B$ be a d. in F, if $B = A$ and F is a minimal cover, then $Y = R-A$ (Y is a superkey); if $B \neq A$ then $B \in R-A$ and so B is prime

3. if $XA \in \rho$
   as F is a minimal cover, it cannot exist a f.d. $X' \to A$ in F s.t. $X' \subset X$ and hence X is a key of XA; let $Y \to B$ be any d. in F s.t. $YB \subseteq XA$, if $B = A$ then as F is a minimal cover, $Y = X$ (Y is a superkey), if $B \neq A$ then $B \in X$ (B is prime)

Uniqueness test

we compute the intersection of the sets $X = R-(W-V)$ with $V \to W \in F$ (for $\forall\, V \to W \in F$), if the intersection determines R, then it's the only key to R

↓ with its closure ↳ $X \to R \in F^+ \iff R \subseteq X_F^+$

- internal difference excludes reflexive dependencies
- external difference excludes dependent attributes

ex.

$X = ABDE$
$X = ACDE$
$X = ABCD$
$\Big\} \cap = AD$

$(AD)^+ = AD \neq R$ → more than one key

Definitions

relation schema

→ a relation name R with a set of attribute names: $R(A_1, \dots A_n)$

instance and legal instance

→ given a relation schema R, an instance of R is a set of tuples on R

→ a legal instance satisfies all the f.d. of the set F. Given R and $X \to Y$ an instance r of R satisfies $X \to Y$ if: $\forall t_1, t_2 \in r : t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$

functional dependency → semantic link between 2 sets of attributes X and $Y \in R$

closure of F (F^+)

→ set of f.d. that are satisfied by each legal instance of R ($F \subseteq F^+$)
$X \to Y \in F^+ \iff \forall A \in Y,\ X \to A \in F^+$

→ set F^+ that includes F and is closed under the Armstrong Axioms, F^+ is found by applying them repeatedly until no more f.d. can be derived from F.

Armstrong Axioms

1. if $Y \subseteq X \in R$ then $X \to Y \in F^A$ reflexivity
2. if $X \to Y \in F^A$ then $XZ \to YZ \in F^A$, for each $Z \in R$ augmentation
3. if $X \to Y \in F^A$ and $Y \to Z \in F^A$ then $X \to Z \in F^A$ transitivity

Keys

→ a subset K of attributes of R is a key of R if $K \to R \in F^+$ and there is no proper subset $K' \subseteq K$ s.t. $K' \to R$

→ superkeys: set of attributes with the property that no 2 tuples in any legal instance r will have $t_1[S] = t_2[S]$, a key is a superkey but the removal of any attribute will cause K not to be a superkey anymore, we say it is minimal

→ primary keys

→ candidate keys

Prime attributes

→ if an attribute is a member of some candidate key

→ if it is not: non prime attribute

X_F^+

- $X_F^+ = \{A \mid X \to A \in F^A\}$
- $X \subseteq X_F^+$
- all those attributes that are functionally determined by X, by possibly applying A.A.

3NF

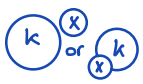
- R_i is in 3NF if $\forall X \to A \in F^+$, $A \notin X$, A belongs to a key (prime) OR X contains a key (superkey)
- all trivial dependencies must not be checked
- R is in 3NF iff there are no partial d. and no transitive d.

Partial and transitive f.d.

- partial d.: when a dependency is a consequence of another

$X \to A \in F^+ \mid A \notin X$ is a p.d. on R if A is not prime and X is properly contained in a key of R ($X - K = \emptyset$)

- transitive d.: when a dependency is a consequence of two functional dependencies



$X \to A \in F^+ \mid A \notin X$ is a t.d. on R if A is not prime and for each key K of R we have that X is not properly contained in K ($X - K \neq \emptyset$) and $K - X \neq \emptyset$

Boyce-Codd Normal form

- a relation is in BCNF when every determinant in it is a superkey (a key is also a s.k.)
- if a relation is in BCNF it is also in 3NF, opposite not true
- can't always decompose in BCNF but always in 3NF

Equivalence

- $F \equiv G$ if $F^+ = G^+$; how to check? $F^+ \subseteq G^+$ and $G^+ \subseteq F^+$, but enough to just check $F \subseteq G^+$ because every f.d. in F can be derived from G by A.A.

- $\pi_{R_i}(F)$: every projection of F that is included by def. in G is a subset of F^+, so F^+ contains G ($G \subseteq F^+$) and the lemma about the closure inclusion (4°) says that if $G \subseteq F^+$ then $G^+ \subseteq F^+$

G

- $G = \bigcup_{i=1}^{k} \pi_{R_i}(F)$

G^+

- check def. for F^+

ρ preserves F

- ρ preserves F if $F \equiv G = \bigcup_{i=1}^{k} \pi_{R_i}(F)$ where $\pi_{R_i}(F) = \{X \to Y \in F^+ \mid XY \subseteq R_i\}$
- G is a set of f.d., each $\pi_{R_i}(F)$ is a set of f.d. obtained by projecting F onto the subschema R_i (taking all dependencies in F^+ that have all the attributes in R_i)

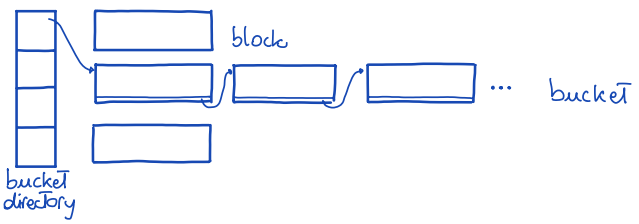
Minimal covers

→ a set of f.d. G, equivalent to F, such that:

1. for each f.d. in G the dependent is a singleton (each dependent is non-redundant)
2. for each dependency $X \to A$ in G there not exists $X' \subset X$ s.t. $G \equiv (G - \{X \to A\}) \cup \{X' \to A\}$ (each determinant is non-redundant)
3. there not exists any $X \to A$ in G such that $G \equiv G - \{X \to A\}$ (each dependency is non-redundant)

Physical Organization

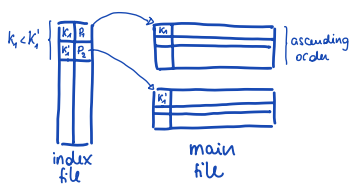
Hash



in hash file org. data is stored based on the result of a hash function applied to the key value of each record. The hash function is applied to the search key and the record is found at the index that results from the hash function. Constant-time access to individual records. $\lceil \frac{\text{Blk\_per\_Bucket}}{2} \rceil$ → linear search along blocks in bucket

in heap file org. records are stored in no particular order and are retrieved by a full table scan. There is no index so every blk in the table must be accessed to find a particular record. $\frac{n}{2}$ blk → linear search

ISAM (indexed sequential access method)



- file divided into intervals or partitions
- each interval is represented by index entry (in index file)
  - search key value of first record in interval
  - pointer to physical position of first record in interval

- data file is ordered on unique key, index is defined over this unique search key
- search time: $\log_2(nbi) + 1$ with $nbi \ll n$ (nbi = # of blocks in index) (index based search)
  - ↳ access block in main file
- binary search

B-tree

- every node in B-tree except the root (at least 2 children, up to m) and leaf nodes have at least (min.) $m/2$ children (max = m)
  - this controls the height of the tree, to be better than m-way search trees, reduce cost of search (also for dynamism)
- every node has: a set of search key values, a set of tree pointers to children, a set of data pointers that refer to data records, or blocks with data records that correspond to search key values
- access is made recursively from the root, insertion and deletion may cause some problems due to the node being full or being smaller than the min. requested
- average cost for searching a B-tree is: cost $\leq$ tree height $-1+1$
  - root node in RAM
  - accessing the data file (not needed if index contains the data)
- min. height (best case): $\lceil \log_m (N+1) - 1 \rceil$
  - N = # nodes
  - m = max # children for node
- max. height (worst case): $\lfloor \log_d (N+1)/2 \rfloor$
  - d = min # children a node can have ($m/2$)

Binary search

- in a normal b.s. on a sorted array you perform the search on the array elements, in ISAM you use the index to guide the search
- in a normal b.s. you access the array elements based on their indices, in ISAM you use the index to access portion of record but other operations may be needed to actually retrieve the record from the main file.
- in normal b.s. the array does not change size or structure, in ISAM indexes are often dynamic and may need to be updated to maintain efficiency, this is because records can be inserted or deleted affecting the index structure