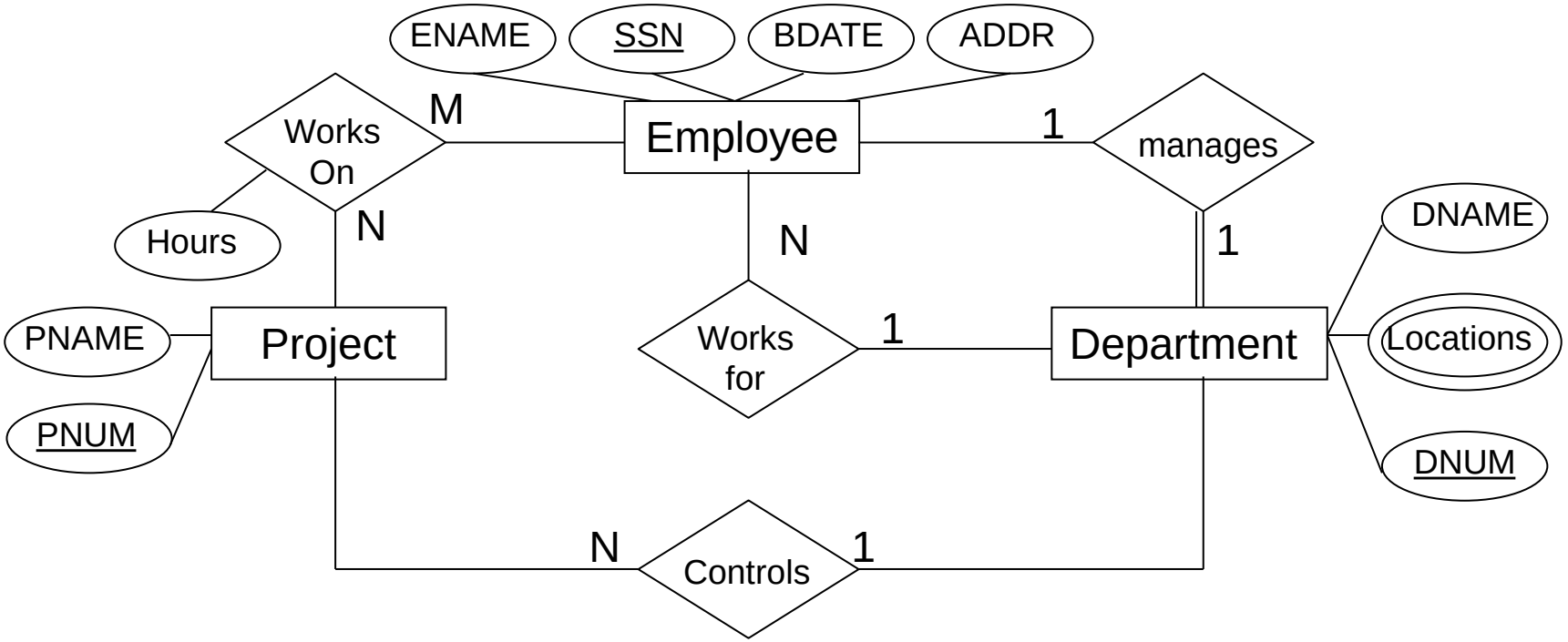
Functional Dependencies and Normalization for Relational Databases

Βάσεις Δεδομένων
Πολυτεχνείο Κρήτης

Relational Schema Design

- Semantics of attributes
- Reducing the redundant values in tuples
- Reducing the null values in tuples
- Disallowing spurious tuples

Semantic of attributes



Semantic of Attributes (cont)

Employee

EName	SSN	BDATE	ADDR	DNum

Department

DNAME	DNumber	DMGRSSN

Project

Pname	Pnumber	PLOCATION	DNUM

WorksOn

SSN	PNumber	HOURS

DeptLocations

DNUM	DLOCATION

- **Guideline:** Design a relation schema in such a way that is easy to explain its meaning. Typically, this means that we should **not** combine attributes from multiple entity types and relationship types into a single relation

Relational Database Instance

EMPLOYEE

ENAME	SSN	BDATE	ADDR	DNO
John Smith	1	1/4/70	Rice 321	1
James Borg	2	4/5/72	Space 2	1
Nick Geb	3	23/6/80	Stone 101	2
Ann Wallace	4	14/9/79	Rich 32	3

DEPARTMENT

DNAME	DNO	…
Research	1	…
Administration	2	…
Headquarters	3	…

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM
ProductX	1	Bellaire	3
ProductY	2	Sugarland	1
ProductZ	3	Houston	2
ProductG	4	Stafford	1

DEP. LOCATIONS

DNUMBER	DLOCATION
1	Houston
2	Stafford
3	Belaire
3	Houston

WORKS_ON

SSN	PNUMBER	HOURS
1	1	32
1	2	10
2	3	22
3	3	15
4	4	5

Redundant Information

- Insertion / Deletion / Update Anomalies

EMP_DEPT

ENAME	SSN	BDATE	ADDR	DNO	DNAME	…
John Smith	1	1/4/70	Rice 321	1	Research	…
James Borg	2	4/5/72	Space 2	1	Research	…
Nick Geb	3	23/6/80	Stone 101	2	Administration	…
Ann Wallace	4	14/9/79	Rich 32	3	Headquarters	…

EMP_PROJ

SSN	PNUMBER	HOURS	ENAME	PNAME	PLOCATION
1	1	32	John Smith	ProductX	Belaire
1	2	10	John Smith	ProductY	Sugarland
2	3	22	James Borg	ProduxtZ	Houston
3	3	15	Nick Geb	ProduxtZ	Houston
4	4	5	Ann Wallace	ProductG	Stafford

•**Guideline:** Design the relation schemas so that no insertion, deletion, or modification anomalies occur in the relations. If any anomalies are present, note them clearly so that the update programs will operate correctly

Null Values in Tuples

- Nulls can have multiple interpretation
  - The attribute does not apply to this tuple
  - The attribute value for this tuple is unknown
  - The value is known but absent, has not been recorded yet

- Group many attributes in a “fat” relation
  - This may lead to many null values in tuples
    - Waste of space
    - Problems with joins and aggregate operations

- **Guideline:** As far as possible, avoid placing attributes in a relation whose values may be null. If nulls are unavoidable, make sure that they apply in exceptional cases only and do not apply to a majority of tuples in the relation

Spurious Tuples

EMP_LOCS

ENAME	PLOCATION

EMP_PROJ1

SSN	PNUMBER	HOURS	PNAME	PLOCATION

ENAME	**LOCATION**
Nikos Pappas	Chania
Fotis Kazasis	Chania

SSN	PNUMBER	HOURS	PNAME	PLOCATION
1	1	5	project1	chania
2	1	10	project1	chania

ENAME	LOCATION	SSN	PNUMBER	HOURS	PNAME
Nikos Pappas	Chania	1	1	5	project1
Nikos Pappas	Chania	2	1	10	project1
Fotis Kazasis	Chania	1	1	5	project1
Fotis Kazasis	Chania	2	1	10	project1

- **Guideline:** We should design relation schemas so that they are joined with equality conditions on attributes that are either primary keys or foreign keys in a way that guarantees that no spurious tuples are generated.

Functional Dependencies

- A functional dependency (FD) on a Relational Schema R is a constaint of the form X->Y, where X,Y are set of attributes used in R.
- If r is a relation instance of R is said to satisfy this FD, if for every pair of tuples t & s in r, t & s agree on all attribute in X then t & s agree on all attributes in Y.

Note:

- If a constaint on R states that there cannot be more than one tuple with a given X-value in any relation instance r(R) – that is, X is a **candidate key** of R – this implies that X->Y for any subset of attributes Y of R.
- If X->Y in R, this does not say whether or not Y->X in R.

Functional Dependencies (cont)

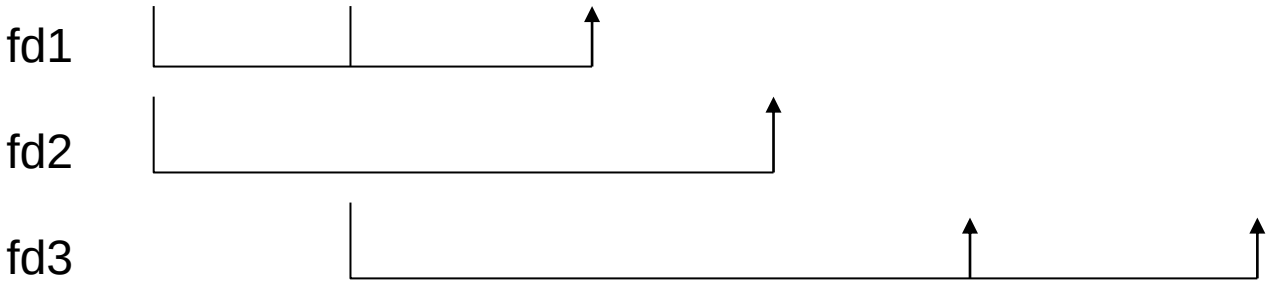
- A functional dependency is a property of the meaning or semantics of the attributes in a relation schema R.

- Relation instances r that satisfy the FD constraints specified on the attributes of R are called **legal relation instances**

Functional Dependencies (example)

EMP_PROJ

SSN	PNUMBER	HOURS	ENAME	PNAME	PLOCATION



- Fd1: {SSN,PNUMBER} -> HOURS
- Fd2: SSN->ENAME
- Fd3: PNUMBER->{PNAME,PLOCATION}

Functional Dependencies (example)

TEACHER	COURSE	TEXT
Smith	Data Structures	Bartam
Smith	Data Management	Al-Tour
Hall	Compilers	Hoffman
Brown	Data Structures	Augenthaler

- **TEACHER -> COURSE ?**
- **TEXT -> COURSE ?**

Inference Rules for Functional Dependencies

- Functional dependency set F
- An FD X->Y is inferred from a set of dependencies on R, if X->Y holds in every relation instance r that is regal instance of R
- Inference rules
- Closure of F , F^+

Example:

F:

SSN-> {ENAME,BDATE,ADDR,DNUMBER}
DNUMBER->{DNAME,DMGRSSN}

Infer:

SSN->{DNAME,DMGRSSN}
SSN->SSN
DNUMBER->DNAME

Inference Rules

- **Reflexive rule** **Y $\subseteq$ X then X->Y**
- **Augmentation rule** **X->Y then XZ->YZ**
- **Transive rule** **{X->Y,Y->Z} then X->Z**
- Decomposition rule {X->YZ} then X->Y
- Union rule {X->Y,X->Z} then X->YZ
- Pseudotransitive rule {X->Y,WY->Z} then WX->Z

- **The first 3 IRs are sound and complete**
- **Armstrong axioms**

Discover the dependencies in a Relation Schema

- First specify the set of functional dependencies F that can easily be determined from the semantics of the attributes in the relations
- Determine each set of attributes X that appears as a left hand side of some FD in F.
  Use Amstrong axioms to determine the set of all attributes that are dependent on X.

- The closure of X under F is the set X^+ of attributes that are functionally determined by X.

Determining the Closure (X^+)

- Algorithm:

$X^+ = X;$

repeat

$oldX^+ = X^+;$

for each functional dependency $Y -> Z$ *in* F *do*

if $Y \subseteq X^+$ *then* $X^+ = X^+ \cup Z;$

until $(oldX^+ = X^+);$

Equivalence of Sets of Functional Dependencies

- A set of functional dependencies F is said to cover E, if every FD in E is also in F^+; that is every dependency in E can be inferred from F

- Two sets of functional dependencies E and F are said to be equivalent if $E^+ = F^+$

Minimal Sets of Functional Dependencies

- A set of functional dependencies F is minimal if it satisfies the following conditions:

    1. Every dependency in F has a single attribute for its right-hand side
    2. We cannot replace any dependency X->A in F with a dependency Y->A, where Y is a proper subset of X, and still have a set of dependencies that is equivalent to F
    3. We cannot remove any dependency from F and still have a set of dependencies that is equivalent to F

Computation of a minimal Cover

Input: a set of FDs F

Output: G, a minimal cover of F

Step 1: G=F, where all FDs are converted to use singleton-attributes on the right-hand side

Step 2: Remove all redundant attributes from the left-hand sides of FDs in G

Step 3: Remove all redundant FDs from G

return G

Normal Forms Based on Primary Keys

- Normalization process
  - As first proposed by Codd (1972), takes a relation schema throughy a series of tests to certify whether or not it belongs to a certain normal form.
- Normalization of data can be looked on as a process during which unsatisfactory relation schemas are decomposed by breaking up their attributes into smaller relation schemas that possess desirable properties

Normalization for RDB

- Normal forms provide DB designers with:
  - A formal framework for analyzing relation schemas based on their keys and the functional dependencies among their attributes
  - A series of tests that can be normalized to any degree. When a test fails, the relation violating that test must be decomposed into relations that will meet the normalization tests.
- The process of normalization through decomposition must also ensure additional properties:
  - The lossless join , which guarantees that the spurious tuple problem does not occur
  - The dependency preservation property, which ensures that all functionall dependencies are represented in some of the individual resulting relations

Normalization for RDB

- Definition of Keys
  - key
  - Candidate key
  - Superkey
  - Primary key
  - Prime attribute
  - Nonprime attribute

First Normal Form

- First normal form disallow multi-valued attributes, composite attributes and their combinations.

- It states that the domains of attributes must include only atomic (simple, indivisible) values and that the value of any attribute in a tuple must be single value from the domain of that attribute

First Normal Form (1NF)

DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	33344	{Bellaire, Sugarland, Houston}
Administration	4	98766	Stafford
Headquarters	1	88888	Houston

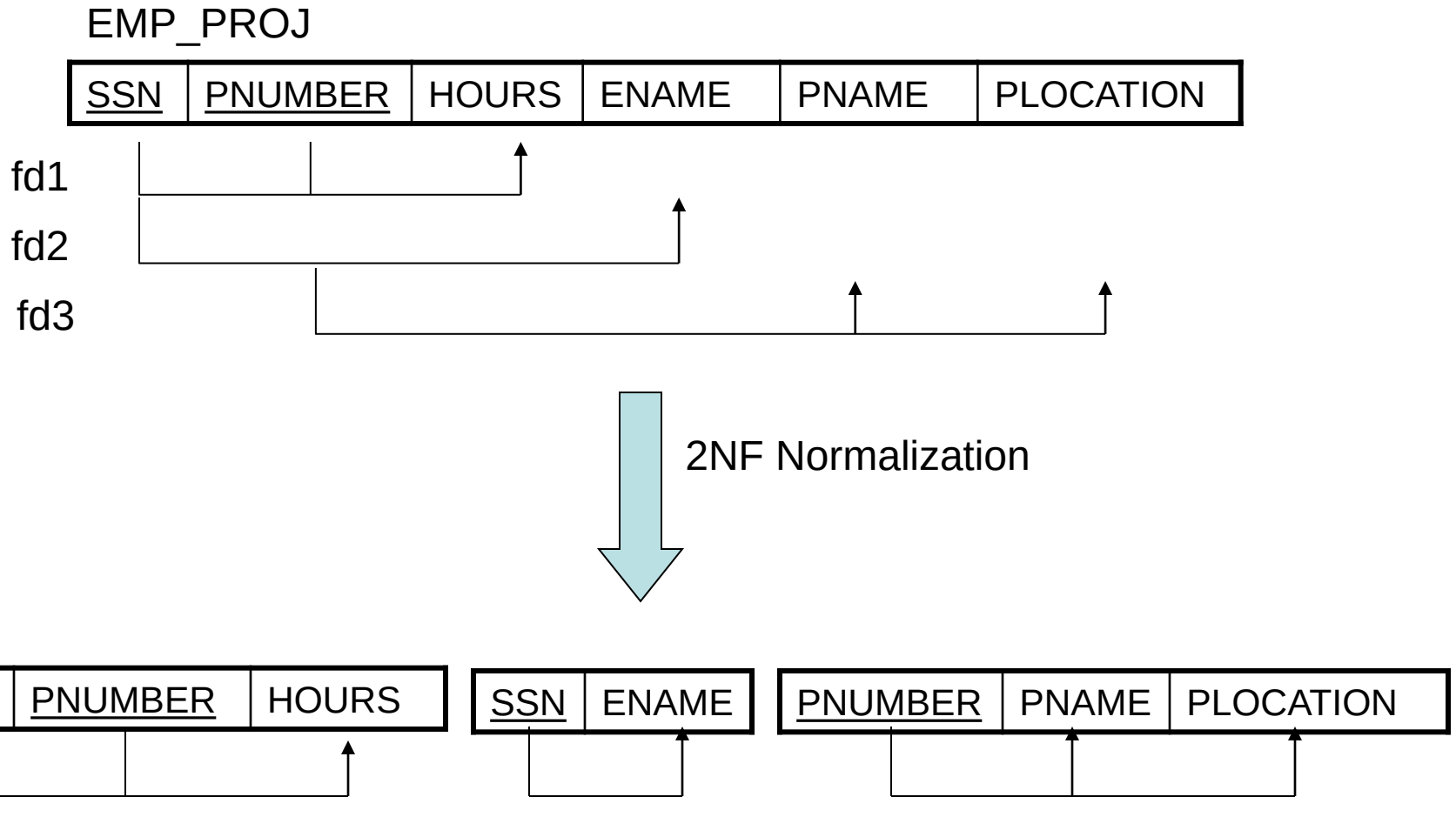
DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	33344	Bellaire
Research	5	33344	Sugarland
Research	5	33344	Houston
Administration	4	98766	Stafford
Headquarters	1	88888	Houston

Second Normal Form (2NF)

- Full functional dependency
  - A functional dependency X->Y is a full functional dependency if removal of any attribute A from X means that dependency does not hold any more
- Partial functional dependency
  - A functional dependency X->Y is a partial dependency if there is some attribute A ε X that can be removed from X and the dependency will still hold

- A relation schema R is in 2NF if every nonprime attribute A in R is fully functionally dependent on the primary key of R

Second Normal Form (2NF) example



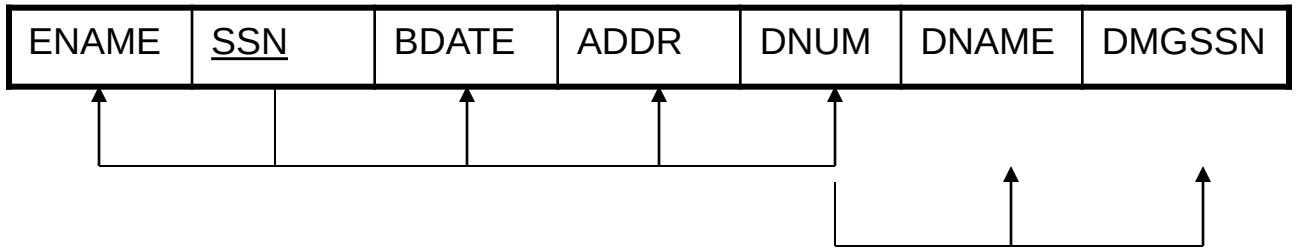
Third Normal Form (3NF)

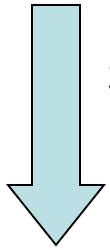
- A functional dependency X->Y in a relation schema R is a transitive dependency if there is a set of attributes Z that is not a subset of any key of R, and both X->Z and Z->Y hold

- A relation schema R is in 3NF if it is in 2NF and no nonprime attribute of R is transitively dependent on a primary key

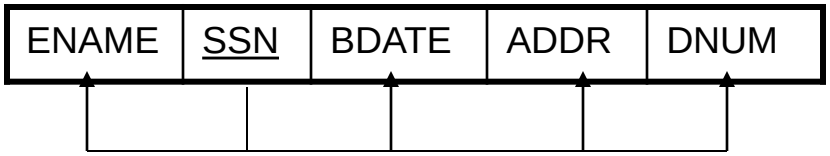
Third Normal Form (3NF) example

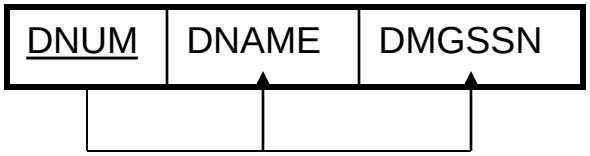
EMP_DEPT





3NF Normalization





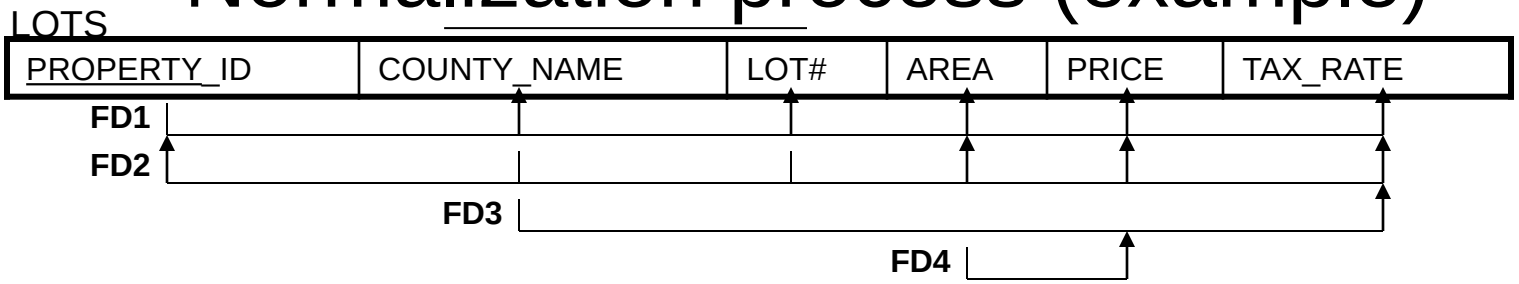
General Definition of NFs

- A relation schema R is in **second normal form (2NF)** if every nonprime attribute A in R is not partially dependent on any key of R.

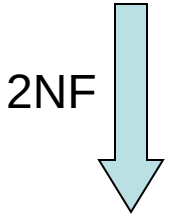
  (A relation schema R is in 2NF if every nonprime attribute A in R is fully functionally dependent on every key of R.

- A relation schema R is in **third normal form (3NF)** whenever a functional dependency X->A holds in R, then either (a) X is a superkey of R or (b) A is prime attribute of R

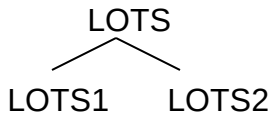
Normalization process (example)

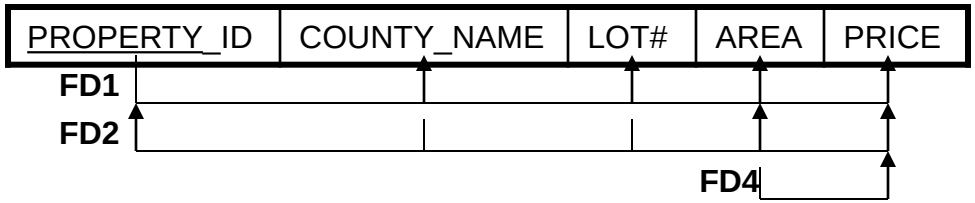


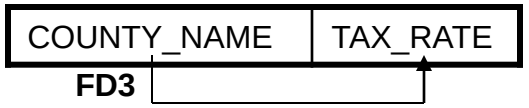
1NF



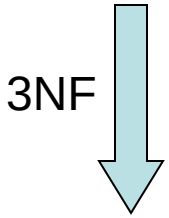
2NF



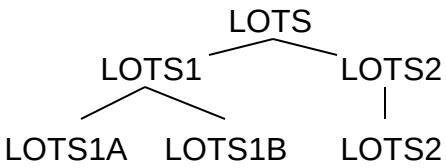


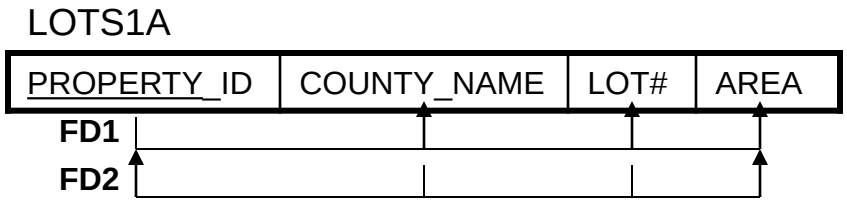


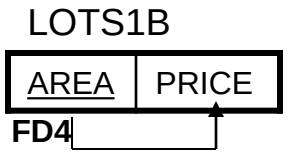
2NF



3NF



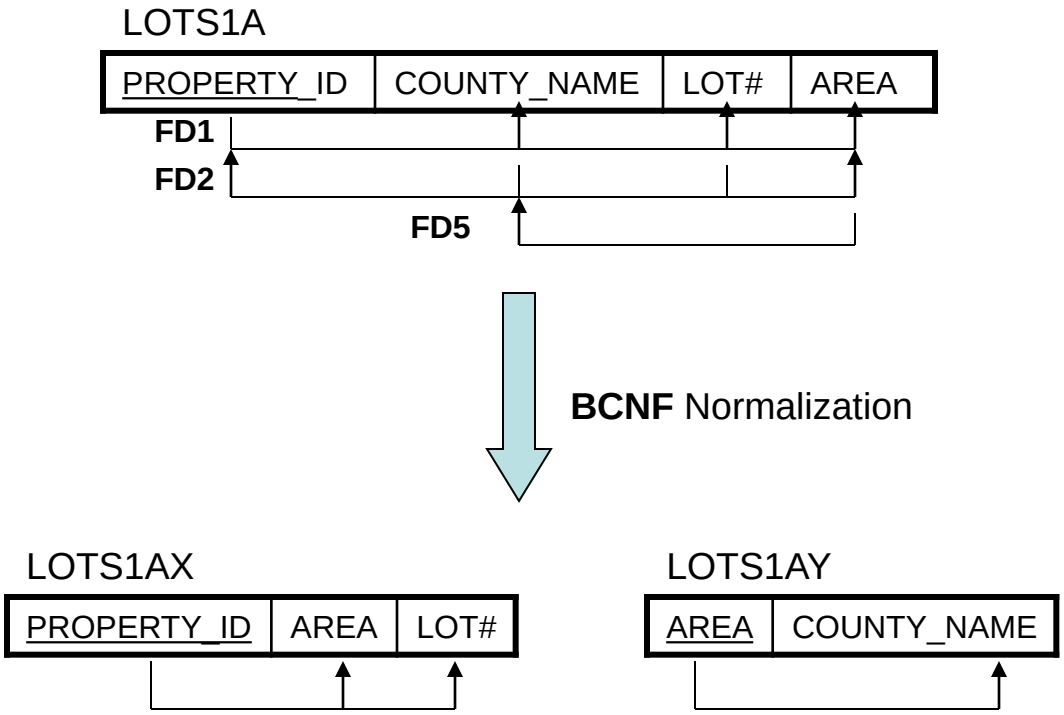




3NF

Boyce-Codd Normal Form (BCNF)

- A Relational schema R is in **Boyce-Codd normal form** if whenever a functional dependency X->A holds in R, then X is a superkey of R



Design of Relational Database Schema

- Universal relation schema R={A1,A2,…,An}
  - Using FDs we decompose the universal schema R into a set of relations D={R1,R2,…,Rm}
  - D is the **decomposition** of R

- Decomposition & **attribute preservation**
- Decomposition & **dependency preservation**
- Decomposition & lossless joins

Design of Relational Database Schema (cont)

- **Decomposition & attribute preservation**
  - Each attribute in R must appear in at least one relation R_i
- **Decomposition & dependency Preservation**
  - Given a set of FDs F on R, a projection of F on Ri, $\pi_F(R_i)$, is the set of dependencies X->Y in F^+ such that the attributes in X U Y are all contained in R_i.
  - A decomposition D={R_1,R_2,…,R_m} of R is **dependency preserving** with respect to F if the union of projections of F on each R_i in D is equivalent to F that is
    $((\pi_F(R_1))U\ldots U(\pi_F(Rm)))^+ = F^+$
- **Decomposition & lossless joins**
  - A decomposition D={R_1,R_2,…,R_m} of R has the **lossless join property** with respect to the set F on R if for every relation instance r of R that satisfies F, the following holds:
    $\bowtie(\pi_{<R1>}(r),\ldots,\pi_{<R2>}(r)) = r$

- An easy test: R decompose to R1,R2. Decomposition is lossless if

$$R1 \cap R2 -> R1 \quad or \quad R1 \cap R2 -> R2$$

Lossless join decomposition into BCNF relations

- Algorithm:

Step 1: set D ← {R};
Step 2: while there is a relation schema Q inD that is not in BCNF do
begin
choose a relation schema Q in D that is not in BCNF;
find a functional dependency X->Y in Q that violates BCNF;
replace Q in D by two schemas (Q-Y) and (XUY)
end;

Lossless join & dependency-preserving decomposition into 3NF relations through schema synthesis

- Algorithm:

Step 1: find a minimal cover G of F;

Step 2: for each left-hand side X that appears in G

create a relation schema $\{X \cup A_1 \cup A_2 \ldots \cup A_m\}$ where $X \rightarrow A_1, X \rightarrow A_2, \ldots, X \rightarrow A_m$ are all the dependencies in G with X as left-hand side;

Step 3: place all remaining (unplaced) attributes in a single relation schema;

Step 4: if none of the relation schemas contains a key of R, create one more relation schema that contains attributes that form a key R;

Design Process

- Determination of all functional dependencies among the database attributes
- Non-deterministic algorithms (minimal cover)

- Top-down database design using ER model
- Combine the two approaches

TUNING ISSUES: To decompose or not to decompose

- Decomposition:
  - Makes answering complex queries less efficient
  - Makes answering simple queries more efficient
  - Makes simple update transactions more efficient
  - Can lower storage space demands