RELATIONAL DATABASE DESIGN

- "Good" database design - Avoiding anomalies
- Functional Dependencies
- Normalization & Decomposition Using Functional Dependencies
- 1NF - Atomic Domains and First Normal Form
- 2NF - Partial Dependencies and Second Normal Form
- 3NF - Transitive dependencies and Third Normal Form
- 4NF - Multi-valued Dependencies and Fourth Normal Form
- 5NF - Decomposition and non loss-less join
- Benefits of Normalization

AVOIDING ANOMALIES - DATABASE CONSISTENCY

How to avoid inconsistent/anomalous state in Databases

- Integrity Constraints address/avoid anomalies that can occur during the Database Manipulation stage
  - e.g. Referential integrities and foreign keys
- Normalization addresses/avoids anomalies that can occur during the Database Design stage - "good" database design

FUNCTIONAL DEPENDENCY

- Relations in the database should be legal under a given set of **Functional Dependencies (FDs)**

Example:

- ***Name → Telephone-number***
- ***Article-number → Price***

- Attribute B (in relation F) is functionally dependent on attribute A (also in relation F) means:
  - For each value of A, there is a unique value of B
  - Written: A → B
  - Read: A "functionally determines" B
    or B is "functionally dependent" on A

FUNCTIONAL DEPENDENCY - continued

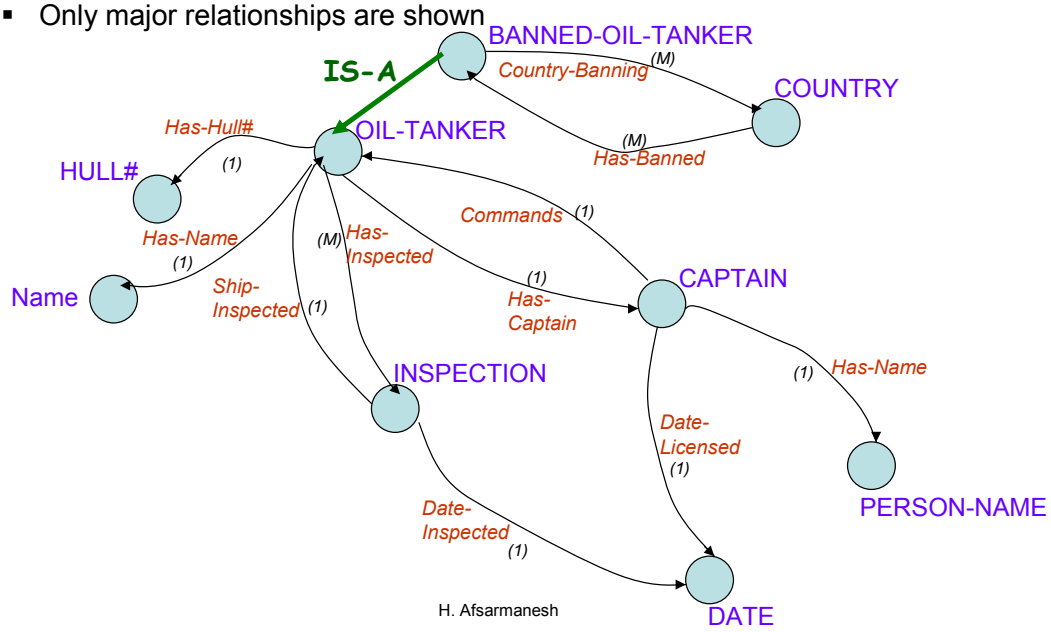
- We can extend the notion of functional dependence to multiple fields
- A1, A2, …, Am → B1, B2, … Bn
  - ***First-name, last-name → Student-ID***
- **Full functional dependence** (*vs. partial functional dependence*) is a functional dependence where there is not a functional dependence from a subset of the A1, A2, …, Am to B1, B2, …, Bn

- **Normalization:** decomposes the relations according to their FDs, to avoid anomalies (e.g. ***insertion, deletion, and update anomalies***)
  - Developed through a number of stages:

1NF, 2NF, 3NF, *BCNF (Boyce Codd Normal Form),* 4NF and 5NF

EXAMPLE O-O DATABASE SCHEMA

- Database of ships with potentially hazardous cargo entering the coastal water of some country (C1)
  - This is a portion of the schema
  - Only major relationships are shown

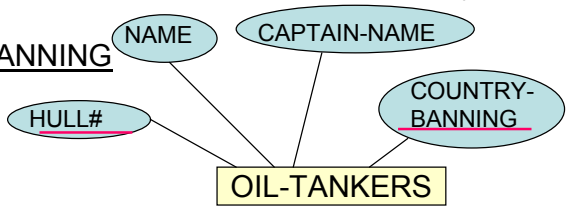


EXAMPLE OF FUNCTIONAL DEPENDENCIES

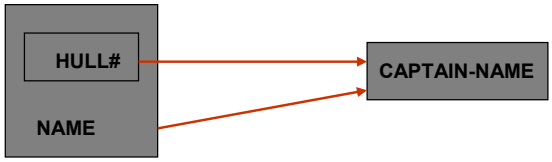
Consider the following relation schema:

OIL-TANKERS (HULL#, NAME, CAPTAIN-NAME, COUNTRY-BANNING)

- primary key: HULL#, COUNTRY-BANNING
- example FDs:
  - HULL# → NAME
  - HULL# → CAPTAIN-NAME *(full functional dependency)*
  - HULL#, NAME → CAPTAIN-NAME *(partial functional dependency)*



FD counterexample: HULL# ↛ COUNTRY-BANNING



*** Dependencies are defined by the database designer, based on the knowledge of the application environment (i.e. they are data-dependent)*

1st NORMAL FORM (1NF)

NORMALIZATION IN RELATIONAL DB SYSTEMS

- Functional dependencies and keys may be used to develop normalization
  - In order to avoid certain types of anomalies

- Normalization steps

1. **FIRST NORMAL FORM**
   - No multi-valued attributes (repeating groups)
     **** Remove multi-valued attributes**

All attributes contain atomic values only

Example in theDatabase of ships environment:
OIL-TANKERS (HULL#, NAME, CAPTAIN-NAME, COUNTRIES-BANNING*)

Modify it to:

- OIL-TANKERS (HULL#, NAME, CAPTAIN-NAME, COUNTRY-BANNING)

Or

- OIL-TANKERS (HULL#, NAME, CAPTAIN-NAME)
  BANS (HULL#, COUNTRY-BANNING)

2nd NORMAL FORM (2NF)

2. SECOND NORMAL FORM

- if 1NF and every attribute not part of the primary key is fully functionally dependent on the primary key.

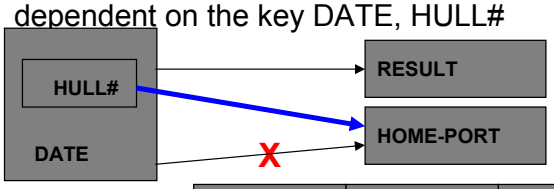
**** Remove partial functional dependencies**

1- Example:
INSPECTIONS (DATE, HULL#, RESULT, HOME-PORT)

2- Functional dependencies:
DATE, HULL# → RESULT
HULL# → HOME-PORT
DATE, HULL# → HOME-PORT

3- the primary key is DATE, HULL#, but HOME-PORT is partially functionally dependent on the key DATE, HULL#

4-



- Causes anomalies:

	DATE	HULL#	RESULT	HOME-PORT
1	d1	h1	Pass	L.A.
2	d2	h1	Fail	L.A.
3	d3	h1	Pass	L.A.

2nd NORMAL FORM (continued)

2. SECOND NORMAL FORM (continued)

5 – anomalies:

Update:

- change HOME-PORT from L.A. to S.F. in the 1st tuple, still key is valid, but the information is wrong since h1 has both L.A. and S.F. as home-port.

Insert:

- h2 with home-port S.D. cannot be added until it is inspected.
- if we add a tuple (d4, h1, pass, S.F.), the system will not catch the inconsistency for the homeport of h1.

Delete:

- delete the last inspection record for h1 and you also loose the information on its home-port

6 – decompose to:

INSEPECTION (DATE, HULL#, RESULT)
SHIP-HOME-PORTS (HULL#, HOME-PORT)

3rd NORMAL FORM (3NF)

3. THIRD NORMAL FORM

- If 2NF and every non-key attribute is non-transitively dependent on the primary key

**** Remove transitive dependencies**

Example:

1- CREW (ID#, HULL#, HOME-PORT)

2- ID# → HULL#
ID# → HOME-PORT
HULL# → HOME-PORT

3- ID# → HULL#
X
HOME-PORT

4- But HULL# is not a candidate key in the CREW relation

CREW

ID#	HULL#	HOME-PORT
C1	h1	L.A.
C2	h1	L.A.
C3	h2	S.F.
C4	h2	S.F.

3rd NORMAL FORM (continued)

3. THIRD NORMAL FORM (continued)

5- Anomalies

Update:

- If C1 starts to work for h2 (HULL#) and you change h1 to h2 then it is inconsistent since h2 is in S.F. but you have it in L.A.

Insert:

- h3 with S.D. cannot be added if there is no ID# for CREW
- If I add (c4 h2 S.D.), it will not be caught by the system as inconsistency for the city home-port of h2

Delete:

- Delete the last tuple for a crew working on a ship, and you also loose the information on the home-port of that ship

6- decompose to

CREW (ID#, HULL#)
SHIP-HOME-PORTS (HULL#, HOME-PORT)

X CREW-HOME-PORTS (ID#, HOME-PORT)
is semantically wrong and has no meaning in real life

4th NORMAL FORM (4NF)

4. FOURTH NORMAL FORM

- Even if a relation is in third normal form, there may still remain some anomalies (problems)
  - o Multi-valued dependency: more general than a functional dependency
  - o A $\rightarrow\!\!\rightarrow$ B (A multi-determines B) if B has a well-defined value (but not necessarily a single value)

**** Remove multi-valued dependencies**

4th NORMAL FORM (continued-1)

4. FOURTH NORMAL FORM (continued)

– Example:

CAPITAIN-NAME →→ LICENSING-COUNTRY

CAPTAIN-NAME	CREW-ID#	LICENSING-COUNTRY
BLY	C1	USA
BLY	C1	GERMANY
BLY	C2	USA
BLY	C2	GERMANY
BLY	C3	USA
BLY	C3	GERMANY
WHITE	C4	ENGLAND
WHITE	C5	ITALY

X Wrong

**** This shows that captain BLY has crew C1, C2, C3, and is licensed from both USA and GERMANY***

QUERY: Find the licensing countries for the captain of the crew c2?

4th NORMAL FORM (continued-2)

4. FOURTH NORMAL FORM (continued)

- Example decomposed relations:

  CAPTAIN-CREW (CAPTAIN-NAME, CREW-ID#)

  CAPTAIN-LICENSES (CAPTAIN-NAME, LICENSING-COUNTRY)

- If all relations are in fourth normal form
  - then, each tuple in each relation consists of one main key, plus some **mutually independent** attribute values

    * then, the key identifies an **object**, and other attribute values describe that object

5th NORMAL FORM

5. FIFTH NORMAL FORM

- Even if a relation is in fourth normal form, there may still be more anomalies
  - o **Non-loss- decomposition:** no loss of information (semantics) must occur with a join after a decomposition
  - o usually, a join after a decomposition returns the original (pre-decomposition) relation
  - o BUT, there may be ***join dependencies***

**** Remove join dependencies**

RA	RB	RC
A1	B1	C2
A2	B1	C1
A1	B2	C1
A1	B1	C1

Lossy decomposition

RA	RB	RB	RC
A1	B1	B1	C2
A2	B1	B1	C1
A1	B2	B2	C1

RA	RC
A1	C2
A2	C1
A1	C1

Join on RB

RA	RB	RC
A1	B1	C2
A1	B1	C1
A2	B1	C2
A2	B1	C1
A1	B2	C1

X Wrong

Join on RA+RC

RA	RB	RC
A1	B1	C2
A2	B1	C1
A1	B1	C1
A1	B2	C1

NORMALIZATION cont.

5. FIFTH NORMAL FORM (continued)

Lossless Decomposition

- Let R be a relation schema
- Let R1 and R2 form a decomposition of R
- This decomposition is a lossless (join) decomposition of R, if at least one of the following functional dependencies holds:
  - o R1 ∩ R2 → R1
  - o R1 ∩ R2 → R2
- Thus R1 ∩ R1 must form a superkey of either R1 or R2

In the previous example: R = (RA, RB, RC)
R1 = (RA, RB)
R2 = (RB, RC)

R1 ∩ R2 = RB

But, RB is not a superkey of either R1 or R2, thus this decomposition is lossy

BENEFITS OF NORMALIZATION - WHY NORMALIZE?

- Avoid anomalies
- Reduce data redundancy (by decompositions)
- Capture some application environment semantics
  - o Key represents the object-ID
  - o Other attributes describe the object

- ❑ **Functional dependencies capture some facts about the application environment**
- ❑ **Normalization allows us to enforce those semantics into the system**

- however, there are many other types of **semantic constraints** that cannot be captured by functional dependencies
  - relational integrity constraints are required

e.g. No employee's salary can be more than his manager's salary