Navigating INFO

presented by:

Tim Haithcoat
University of Missouri Columbia

with materials from:

Environmental Systems Research Institute

Section Four

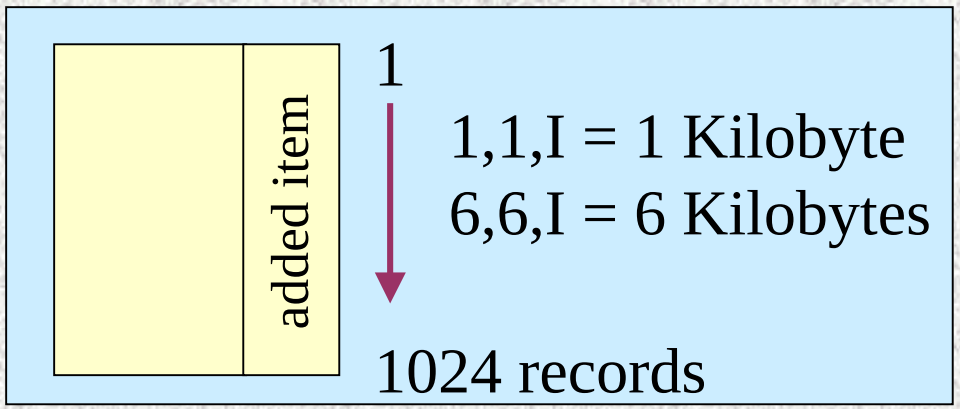
Database Hygiene

Database Hygiene

- Creating and maintaining an efficient relational database
- Modifying existing databases that were poorly defined/implemented
- Subtopics
  - Bits/bytes/item definitions
  - Normalization
  - ARC RELATE command
  - Restructuring tricks - projecting files
  - Minimal output overlays {NOJOIN}

Bits & Bytes & Item Definitions ~ Revisited

- Good database design begins with the proper item definitions (“B” and “F” types as opposed to “I” and “N” types).
- Once a database is built, changing item definitions is very difficult
- Item definition especially important on larger files (each extra byte in an item equals many kilobytes on the file).



Bits & Bytes & Item Definitions ~ Revisited

- Geographic operations (Spatial Joins/Overlays) result in exponentially larger files. Poor item definitions exasperate problems.
- Use “B” and “F” types whenever possible. Know the range of data values for any particular item (use MINMAX procedure on existing databases).

Intro to Normalization

- A computer science term to describe the relationship of data and items within a file
- Understanding and applying normal forms to a relational database is an absolute must in order to have efficient databases.
- Normalization is mostly common sense and knowing your data. It is an intuitive concept.
- Advantages:
  - Flexibility of database is optimized
  - Minimized redundancy; optimized INSERT, UPDATE, and DELETE operations
  - Forces database designer to understand data relationships

Relations

- Relations, or two-dimensional tables of rows and columns, with 6 properties:
  - Each column, termed an attribute, has a unique name
  - The left-to-right order of columns is irrelevant
  - Each attribute entry is single valued; no repeating groups or arrays are allowed
  - Entries in any one column are of the same kind
  - The top-to-bottom order of rows is also irrelevant
  - Each row is unique

Equivalent Relations

5	C	Mary	C	50
12	D	Bob	D	55
8	B	Cathy	B	57
19	G	Jim	G	82
26	L	Ed	C	71
1	K	Ernie	K	61
7	A	Bert	A	52
11	F	Moe	D	55
6	M	Chris	M	49
14	T	Harpo	T	57

LEGAL

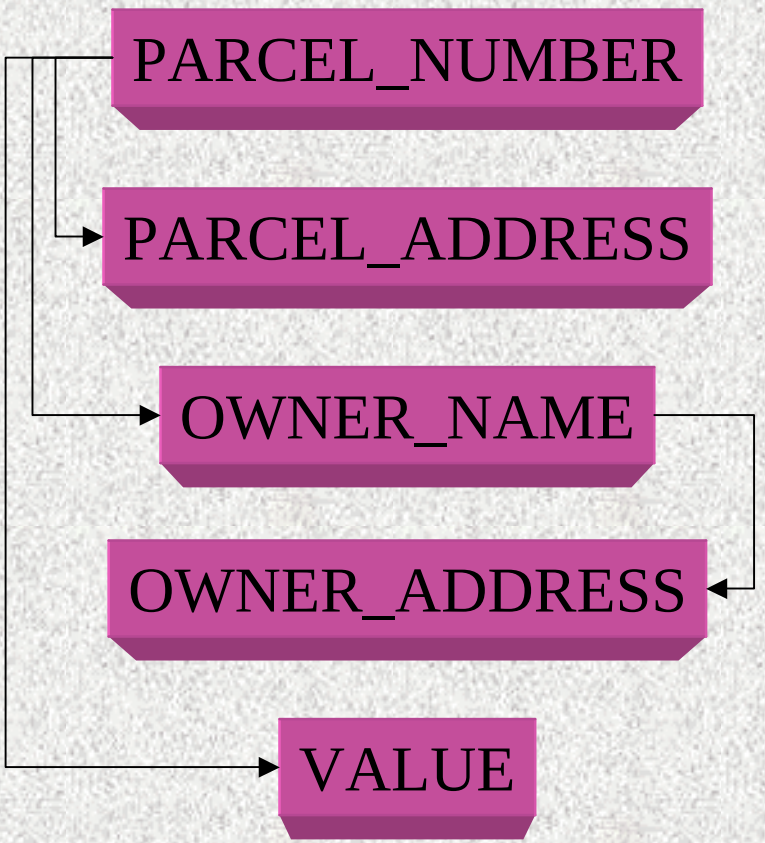
=

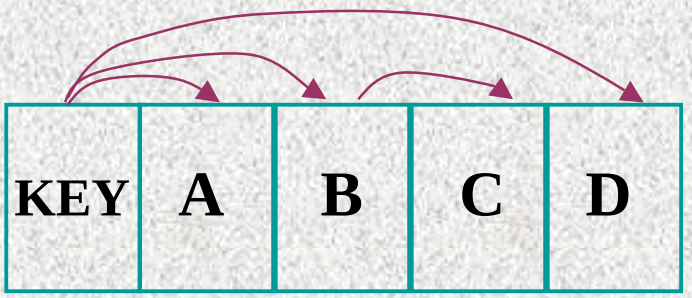
57	B	8	Cathy	B
71	C	26	Ed	L
52	A	7	Bert	A
57	T	14	Harpo	T
49	M	6	Chris	M
55	D	11	Moe	F
82	G	19	Jim	G
55	D	12	Bob	D
50	C	5	Mary	C
61	K	1	Ernie	K

LEGAL

Functional Dependence

PARCEL_OWNER





PARCEL-OWNER

ALLITEMB

PROJBC

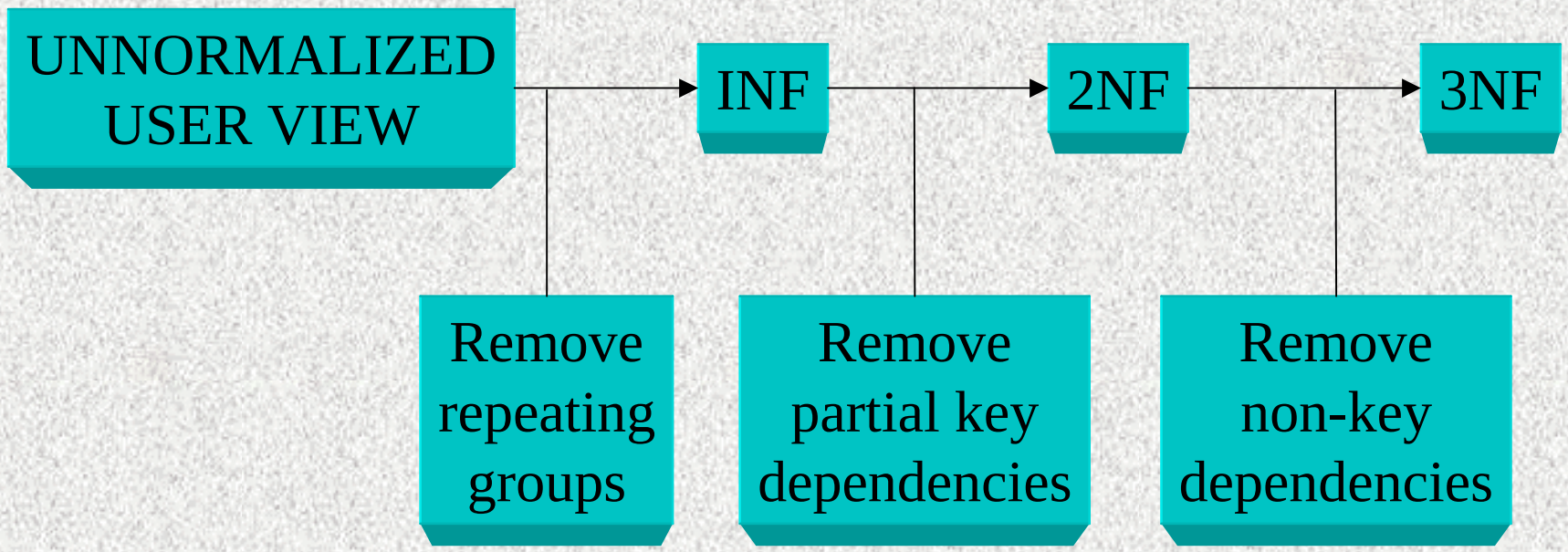
```
DEFINE ALL ITEMS
DEFINE PROJ BC
SEL PARCEL OWNER RO
RELATE ALL ITEMB 1 BY B APP
CALL $NUM1=$NUM + 1
SEL ALL   ITEMB
SORT B

SEL ALLITEMB RO
RELATE PROJBC 1 BY B SUN
CALC $NUM2=$NUM2 + 1
SEL PROJBC
MOVE ____ TO C
SEL PARCEL_ OWNER RO
RELATE PROJBC 1 BY B OR
MOVE C TO $1C
DROP C FROM
     PARCEL_OWNER
```

Functional Dependence

MULTI_OWNER_PARCEL

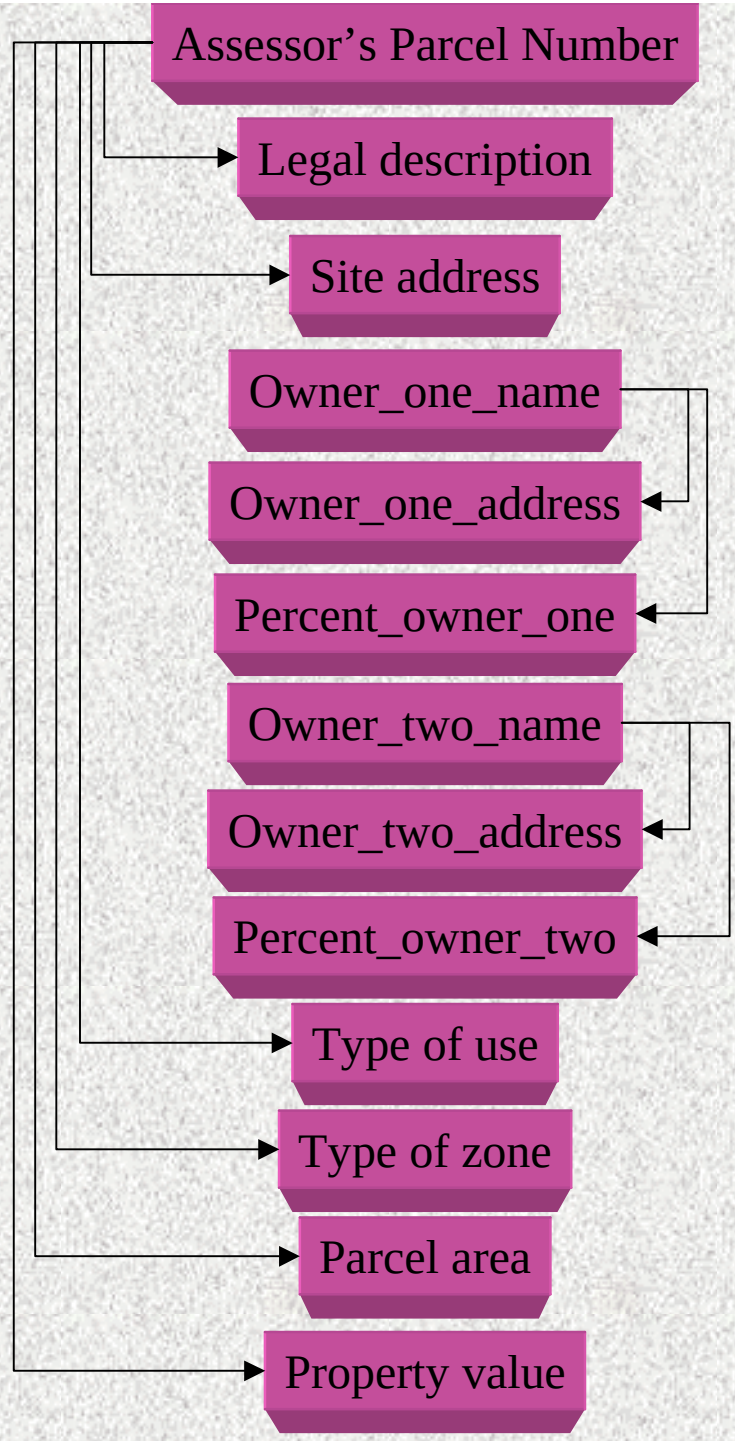
- Assessors Parcel Number
- Legal description
- Site address
- Owner_one_name
- Owner_one_address
- Percent_owner_one
- Owner_two_name
- Owner_two_address
- Percent_owner_two
- Type of use
- Type of zone
- Parcel area
- Property value

Normalization



Normalization

MULTI_PARCEL_OWNER



Normalization

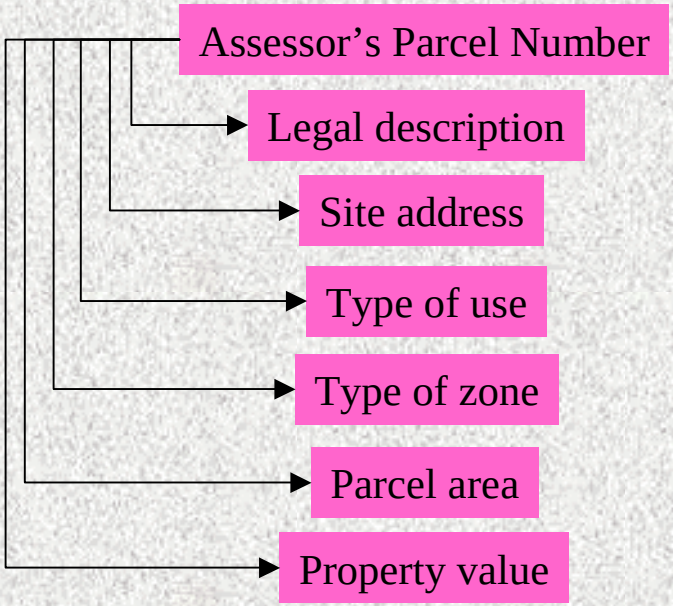
ORIGINAL

MULTI_OWNER_PARCEL

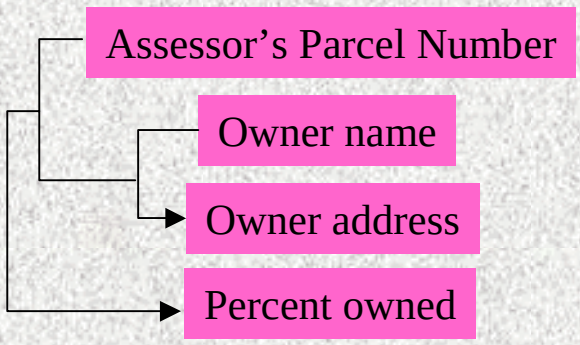
- Assessors Parcel Number
- Legal description
- Site address
- Owner_one_name
- Owner_one_address
- Percent_owner_one
- Owner_two_name
- Owner_two_address
- Percent_owner_two
- Type of use
- Type of zone
- Parcel area
- Property value

INTERIM STEP 1

ASR_PARCEL



OWNERSHIP



Normalization

INTERIM STEP 1

ASR_PARCEL

- Assessor's Parcel Number
- Legal description
- Site address
- Type of use
- Type of zone
- Parcel area
- Property value

OWNERSHIP

- Assessor's Parcel Number
- Owner name
- Owner address
- Percent owned

FINAL STEP 2

ASR_PARCEL

Assessors Parcel Number
Legal description
Site address
Type of use
Type of zone
Parcel area
Property value

OWNERSHIP

Assessor's Parcel Number
Owner name
Percent owned

ASR_OWNER

Owner name
Owner address

Normalization

INTERIM STEP 2

ASR_PARCEL

- Assessor's Parcel Number
- Legal description
- Site address
- Type of use
- Type of zone
- Parcel area
- Property value

OWNERSHIP

- Assessor's Parcel Number
- Owner name
- Percent owned

ASR_OWNER

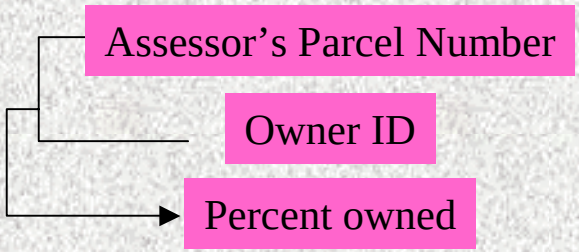
- Owner name
- Owner address

FINAL STEP 3

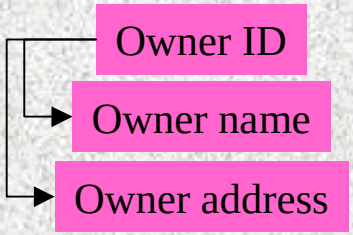
ASR_PARCEL

Assessor's Parcel Number
Legal description
Site address
Type of use
Type of zone
Parcel area
Property value

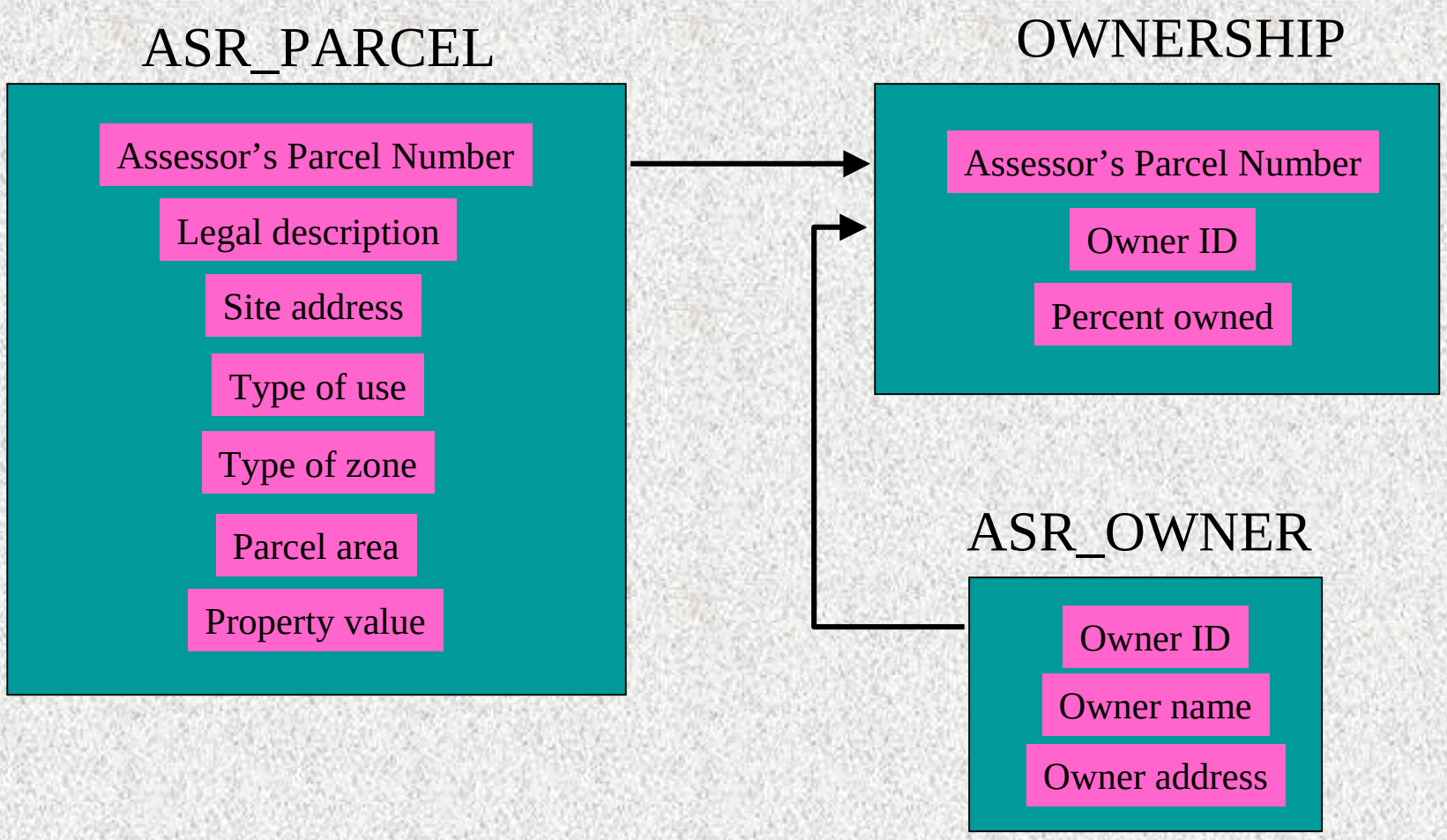
OWNERSHIP



ASR_OWNER



Summary of Normalization



INTRO to NORMALIZATION

- Normalization is:
  - The process of creating relational files which preserve the appropriate dependence and independence of the data items
- A file that has the appropriate dependence and independence of data items is termed normalized
- Three basic normal forms: first, second, third

First Normal Form

UN-NORMALIZED

NODE#	ARC1#	ARC2#	ARC3#	ARC4#	ARC5#
123	10	20	*u*	*u*	*u*
124	27	134	18	*u*	*u*
125	20	15	*u*	*u*	*u*

FIRST NORMAL FORM

NODE#	ARC#
123	10
123	20
124	27
124	134
124	18
125	20
125	15

In the first normal form, every item has a value. In the un-normalized file, the *u* means *"unused"*, which is not a data value (cannot have an *"unused"* arc connecting to a node). The file is un-normalized because very item (e.g., ARC5#) does not have a data value. The normalized file is in (at least) first normal form form because every item has a value.

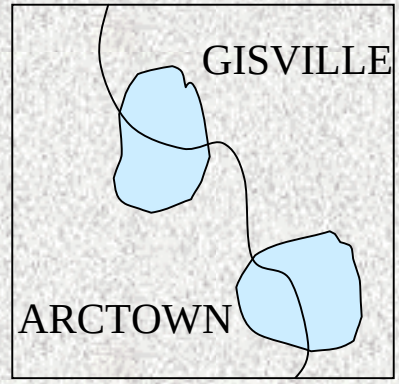
Functional Dependence

- Given a relation R, attribute Y of R is functionally dependent on attribute X of R if and only if, whenever two tuples (records) of R agree on their X-Value, they also agree on their Y-Value.
- That is to say: a data item Y is functionally dependent on another data item X, if every occurrence of X is associated with the value of Y in that record.

CITY.PAT

AREA	PERIMETER	CITY#	CITY-ID	NAME	POPULATION

EXAMPLE MAP



In this relation, NAME is functionally dependent upon CITY-ID. POPULATION is functionally dependent on CITY# (Consider that there may be multiple occurrences of CITY-ID, where a city was split by a county line). CITY_ID is functionally dependent on CITY#.

Second Normal Form

- A table is said to be in second normal form if every non-key item is fully dependent on the primary key item.
- This file is NOT in the second normal form.
- NAME is non-key, yet is not dependent on the primary key, CITY#. NAME and CITY-ID are functionally dependent.

CITY.PAT

AREA	PERIMETER	CITY#	CITY-ID	STATUS	NAME
-23.2	-8989.1	1	0	0	-
4.3	345.1	2	1	1	New Haven
2.4	234.5	3	1	2	New Haven
2.4	234.6	4	2	1	Batville
2.3	321.2	5	3	5	GISville
11.1	7000.0	6	1	5	New Haven

Second Normal Form

CITY.PAT

AREA	PERIMETER	CITY#	CITY-ID	STATUS
-23.2	-8989.1	1	0	0
4.3	345.1	2	1	1
2.4	234.5	3	1	2
2.4	234.6	4	2	1
2.3	321.2	5	3	5
11.1	7000.0	6	1	5

CITY.NAME

CITY-ID	NAME
0	-
1	New Haven
2	Batville
3	GISville

Both these tables are now in Second Normal Form.

Jargon

- Breaking apart a file into normalized relations is called DECOMPOSITION.
- The new files resulting from a decomposition are called PROJECTIONS of the original file.

CITY.PAT

AREA	PERIMETER	CITY#	CITY-ID	STATUS	NAME
-23.2	-8989.1	1	0	0	-
4.3	345.1	2	1	1	New Haven
2.4	234.5	3	1	2	New Haven
2.4	234.6	4	2	1	Batville
2.3	321.2	5	3	5	GISville
11.1	7000.0	6	1	5	New Haven

CITY.NAME

CITY-ID	NAME
0	-
1	New Haven
2	Batville
3	GISville

"CITY.NAME" is a projection of "CITY.PAT". We decomposed "CITY.PAT"

Second Normal Form

FIRST

S#	STAT	CITY	P#	QTY
S1	20	London	P1	300
S1	20	London	P2	200
S1	20	London	P3	400
S1	20	London	P4	200
S1	20	London	P5	100
S1	20	London	P6	100
S2	10	Paris	P1	300
S2	10	Paris	P2	400
S3	10	Paris	P2	200
S4	30	Athens	P2	200
S4	30	Athens	P4	300
S4	30	Athens	P5	400

- This table is NOT in second normal form. The items STAT and CITY (both NON-KEY) are functionally dependent on s#. What is the primary key?

Second Normal Form

SECOND

S#	STAT	CITY
S!	20	London
S2	10	Paris
S3	10	Paris
S4	30	Athens

SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

These relations are in second normal form, because very non-key item is fully dependent on the primary key item. QTY in relation SP is dependent on the S#,P# key. STAT and CITY are dependent on S# in SCEOND. (But what is CITY *really* dependent on?)

Date, C.J., An introduction to database systems, Addison-Wesley Systems Programming Series, 1981.

Third Normal Form

A relation (table) R is in third normal form (3NF) if and only if, for all time, each record of R consists of a primary key value that identifies some entity, together with a set of mutually independent attribute values that describe that entity in some way.

FIRST

S#	STAT	CITY	P#	QTY
S1	20	London	P1	300
S1	20	London	P2	200
S1	20	London	P3	400
S1	20	London	P4	200
S1	20	London	P5	100
S1	20	London	P6	100
S2	10	Paris	P1	300
S2	10	Paris	P2	400
S3	10	Paris	P2	200
S4	30	Athens	P2	200
S4	30	Athens	P4	300
S4	30	Athens	P5	400

SECOND

S#	STAT	CITY
S!	20	London
S2	10	Paris
S3	10	Paris
S4	30	Athens

SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

SC

S#	CITY
S1	London
S2	Paris
S3	Paris
S4	Athens

CS

CITY	STAT
London	20
Paris	10
Athens	30

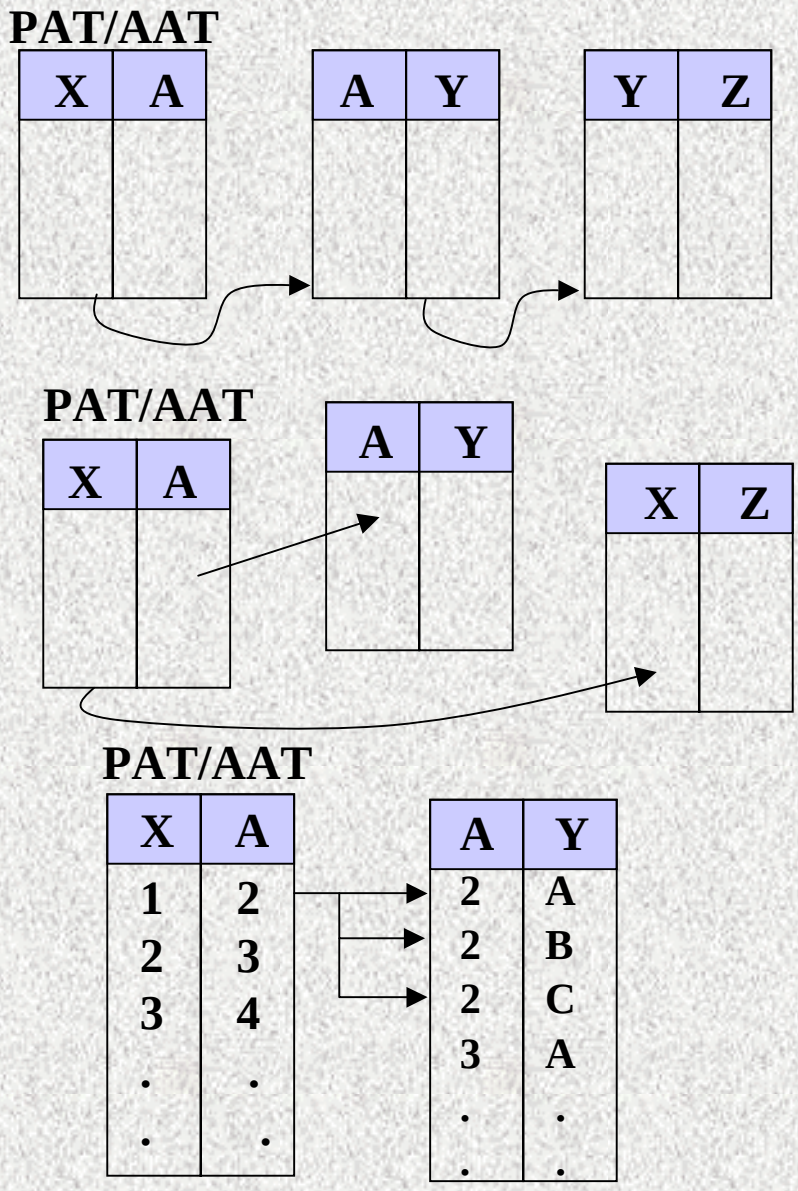
Date, C.J., An introduction to database systems, Addison-Wesley Systems Programming Series, 1981.

REVIEW

- **Normalization is common sense. It requires intuition and knowledge of the database.**
- Know your data well enough to recognize functional dependence of one item to another.
- Decompose all functionally dependent items into separate projections.
- Normalized databases are more flexible and minimize redundancy. They are faster to access that one large, un-normalized file.
- Most relations (files) in an ARC database have COVER# as the primary key.

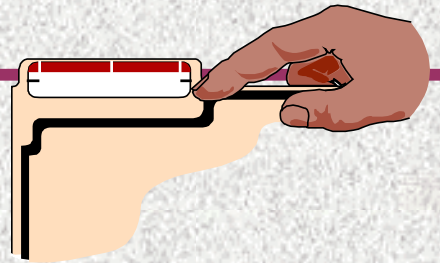
ARC RELATE

allows access to normalized databases



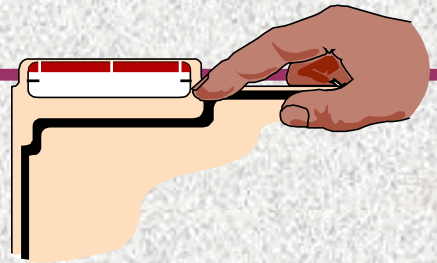
- Chained relates are supported (unless the last file in the chain is a symbol lookup table).
- Multiple relates are supported.
- One to many relationships supported. The returned value of Y might be A,B, or C. Info would return first occurrence - A.

Restructuring Files



- Restructuring includes:
  - Changing item definitions
  - Decomposing files into different projections
- Neither ARC nor INFO provides straight-forward utilities to restructure files.
- Foreign data is usual source of unwieldy files.
  - Supplier joins relations together into one file before transmitting to minimize errors in copying and need for documentation
  - Usual transfer format “card image” which translates to I- and N-TYTE definitions in INFO with lots of blank padding.

Restructuring Files



- Determine which items need their definitions changed.
- Determine range for B-TYPES (2-byte or 4-byte).
- **METHOD 1:**
  - ARC ADDITEM to add temporary item
  - INFO to CALCulate
  - ARC DROPITEM
  - INFO ALTER

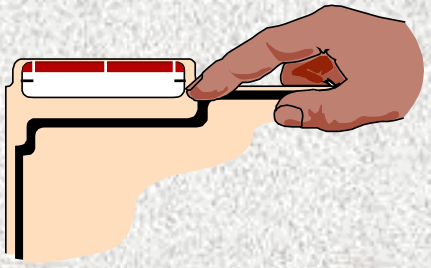
```
ENTER COMMAND>SELROAD.AAT
26119 RECORD(S) SELECTED

ENTER COMMAND>IT
        DATAFILE NAME: R OAD.AAT
        28 ITEMS: STARTING IN POSITION 1
```

COL	ITEM NAME	WDTH	OPUT	TYP	N.DEC
1	FNODE#	4	5	B	-
5	TNODE#	4	5	B	-
9	LPOLY#	4	5	B	-
13	RPOLY#	4	5	B	-
17	LENGTH	4	12	F	3
21	ROAD#	4	5	B	-
25	ROAD-ID	4	8	B	-
*29	MAJOR1	6	6	I	-
*35	MINOR1	6	6	I	-
*41	MAJOR2	6	6	I	-
*47	MINOR2	6	6	I	-
53	OLDCOVID	4	5	B	-
57	QUAD#	5	5	C	-
*62	TETC-ID	5	5	I	-
67	ORIGFNODE	4	5	B	-
71	ORIGTNODE	4	5	B	-
75	STATUS	2	2	C	-
*77	ALK-ID	5	5	I	-
*82	OWNER-NEW	4	4	I	-

* = items we wish to reformat into B-type

Restructuring Files

ALTERNATE METHOD

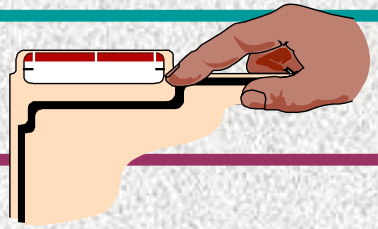


```
ENTER COMMAND> RENAME ARCnnn ROAD.BIGAAT
ENTER COMMAND> SELECT ROAD.BIGAAT
ENTER COMMAND> MODIFY ROAD.AAT
  ( modify instructions)
ENTER COMMAND> SELECT ROAD.AAT
ENTER COMMAND> ITEMS

DATAFILE NAME: ROAD.AAT        28 ITEMS:  STARTING IN POSITION 1
```

COL	ITEM NAME	WDTH	OPUT	TYP	N. DEC
1	FNODE#	4	5	B	-
5	TNODE#	4	5	B	-
9	LPOLY#	4	5	B	-
13	RPOLY#	4	5	B	-
17	LENGTH	4	12	F	3
21	ROAD#	4	5	B	-
25	ROAD-ID	4	8	B	-
*29	MAJOR1	4	6	B	-
*33	MINOR1	4	6	B	-
*37	MAJOR2	4	6	B	-
*41	MINOR2	4	6	B	-
45	OLDCOVID	4	5	B	-
49	QUAD#	5	5	C	-
*54	TETC-ID	4	5	B	-
58	ORIGFNODE	4	5	B	-
62	ORIGTNODE	4	5	B	-
66	STATUS	2	2	C	-
*68	ALK-ID	4	5	B	-
*72	OWNER-NEW	2	4	B	-

* = new item definitions

ALTERNATE METHOD (continued)



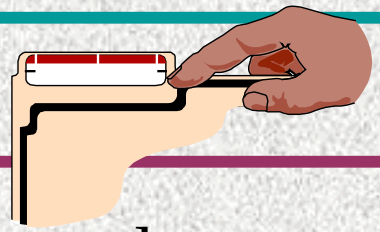
- Move over one item to get 26119 records into ROAD.AAT.

```
ENTER COMMAND> SEL ROAD.BIGAAT
ENTER COMMAND> REL ROAD.AAT1 BY FNODE# APPEND
ENTER COMMAND> CALC $1TNODE# = TNODE#
```

- We could have used a “block move” for a shortcut. A “block move” is where many items are moved over at once.
  - Redefined item”.BLOCK.”, starting in Column 1, 28,28,C
  - Item on both ROAD.BIGAAT and ROAD.AAT

```
ENTER COMMAND> SEL ROAD.BIGAAT
ENTER COMMAND> REL ROAD.AAT 1 BY. BLOCK. APPEND
ENTER COMMAND> CALC $NUM1 + 1
```

ALTERNATE METHOD (continued)



- Complete the transfer of data in an even numbered program section

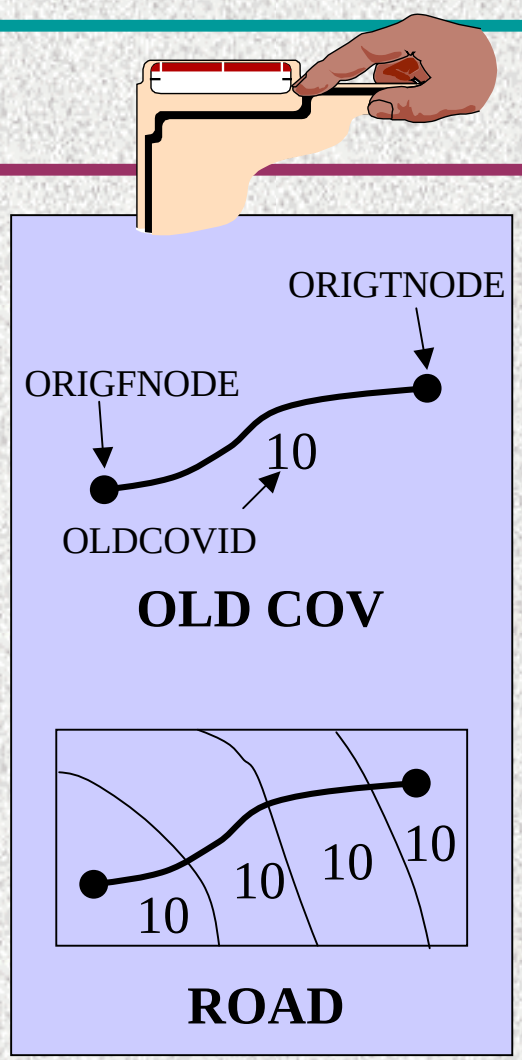
```
ENTER COMMAND> SEL ROAD.BIGAAT
ENTER COMMAND> REL ROAD.AAT1 BY $RECNO LINK
```

- If file was external, first save data to external file

```
ENTER COMMAND> SEL ROAD.BIGAAT
ENTER COMMAND> SAVE (pathname)?AAT INIT
FILE CREATED
ENTER COMMAND> PURGE
This command will delete selected records. OK?> Y
    0 RECORDS SELECTED
ENTER COMMAND? EXTERNAL
Enter Complete File Name of External File (pathname)>AAT
ENTER COMMAND> SEL ROAD.BIGAAT
ENTER COMMAND> DELE ROAD.BIGAAT
This command will erase the specified DF
Do you wish to Continue (Y or n) ? Y
```

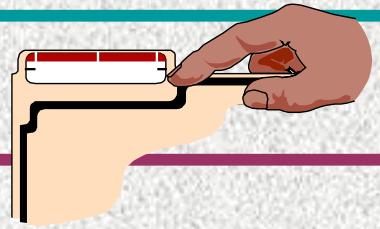
Decompose and Project

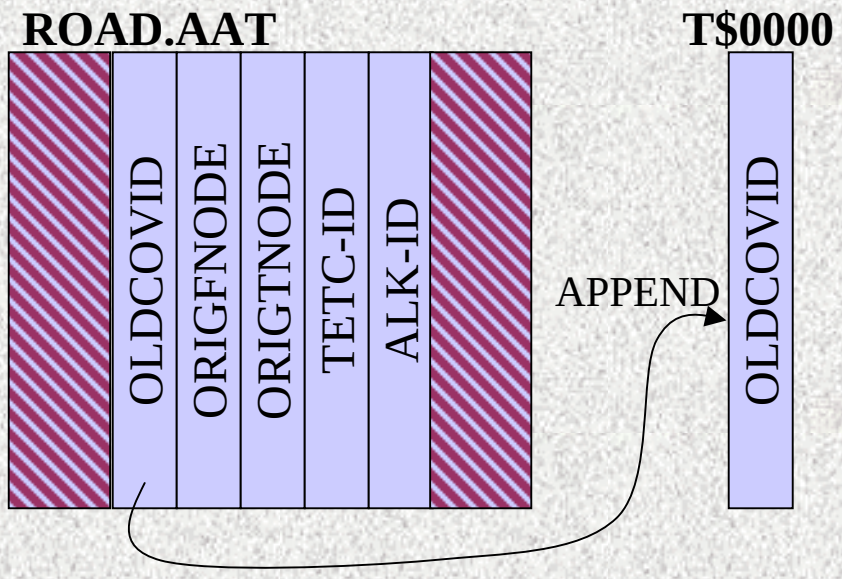
- These items are functionally dependent:
  `OLDCOVID`
  `ORIGFNODE`
  `ORIGTNODE`
  `TETC-ID`
  `ALK-ID`
- This is because the cover ROAD was derived from the cover OLDCOV.
- Items are functionally dependent on the non-key item OLDCOVID. The primary key to ROAD.AAT is ROAD#.
- There are multiple occurrences of OLDCOVID because arcs have been split.



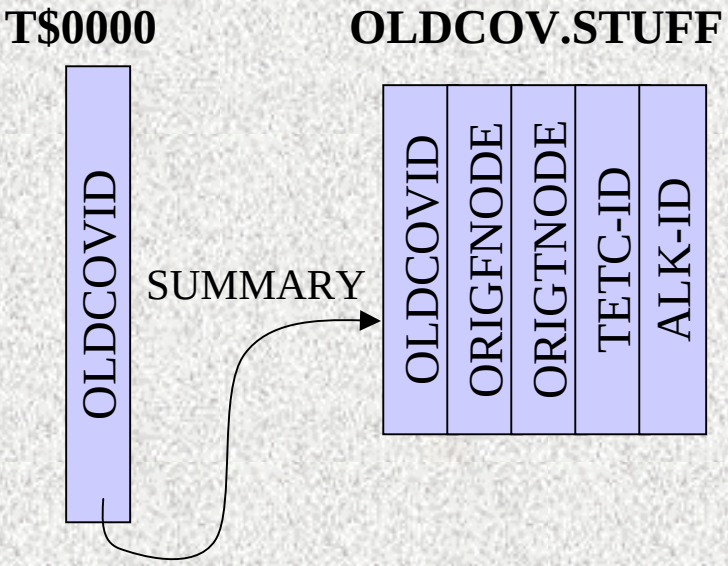
Multiple occurrences of OLDCOVID on coverage ROAD because of arc splitting (perhaps from an overlay).

Decompose and Project



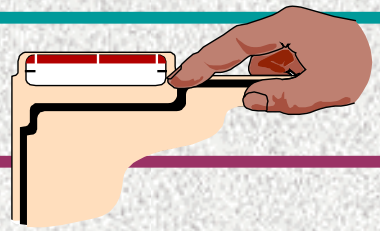


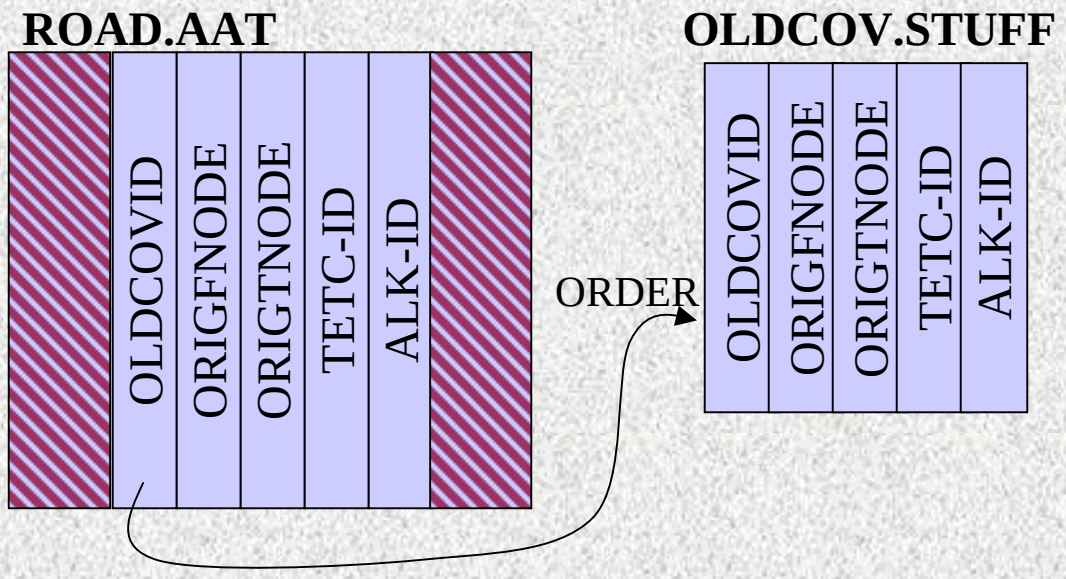
- Define a temporary file, T$0000, and copy OLDCOVID into it.



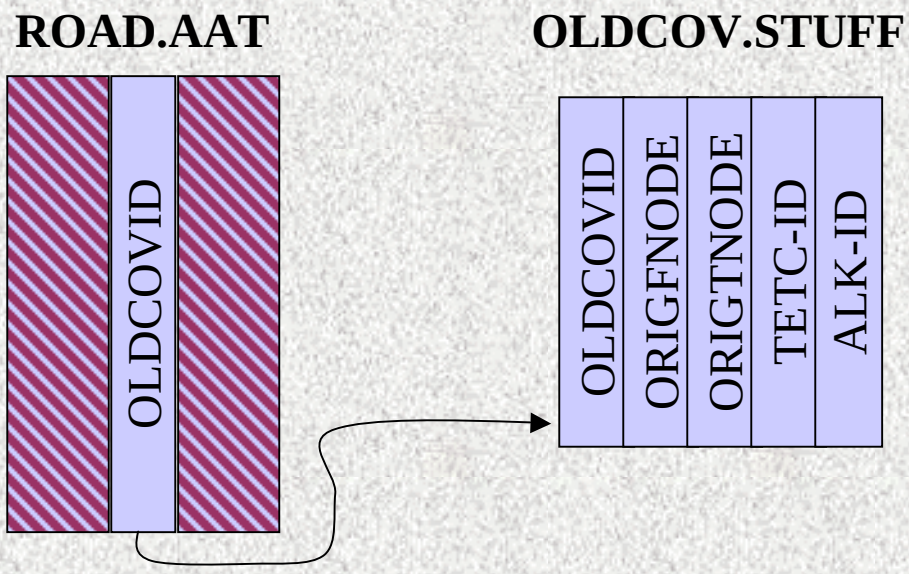
- Sort T$0000 on OLDCOVID. Initialize OLDCOV.STUFF using a summary relate.

Decompose and Project





- Select ROAD.AAT and relate OLDCOV.STUFF by OLDCOVID with *ORDER*. Move over ORIGFNODE, ORIGTNODE, TETC-ID, and ALK-ID.



- Drop the items ORIGFNODE, ORIGTNODE, TETC-ID, and ALK-ID from ROAD.AAT. Use ARC RELATE to access data in OLDCOV.STUFF when needed for ARCPLOT, ARCEDIT, etc.

Minimal Output Overlay

POLY.PAT

AREA	PERIMETER	POLY#	POLY-ID	ITEM1	ITEM2	ITEM3	ITEM4

LOOP.AAT

FNODE#	TNODE#	LPOLY#	RPOLY#	LENGTH	LOOP#	LOOP-ID	RT-ID	NEXT-ARC	CUM-DMNO	CUM-IMPED

ARC IDENITY LOOP POLY OVRCOVER LINE

OVERCOVER.AAT

FNODE#	TNODE#	LPOLY#	RPOLY#	LENGTH	OVRCOVER#	OVRCOVER-ID	LOOP#	LOOP-ID	RT-ID	NEXT-ARC	CUM-DMNO	CUM-IMPED	POLY#	AREA	PERIMETER	POLY-ID	ITEM1	ITEM2	ITEM3	ITEM4

Minimal Output Overlay

- ARC 5.0 supports a NOJOIN option on all overlay commands that makes this method obsolete. Use this method until you have ARC 5.0.
- PRE-ARC 5.0 Method:
  - Use INFO to rename the two input coverages' feature attribute tables to something besides the default AAT or PAT.

```
ENTER COMMAND>DIR LOOP.AAT
TYPE NAME      INTERNAL NAME  NO.RECS LENGTH EXTERNAL
 DF  LOOP.AAT   ARC005DAT        6       44      XX
ENTER COMMAND>RENAME ARC005 LOOPAAT
```

(Do the same for POLY.PAT, renaming it POLYPAT.)

  - Issue the OVERLAY COMMAND.

Overlay - Minimal Output

POLYPAT

AREA	PERIMETER	POLY#	POLY-ID	ITEM1	ITEM2	ITEM3	ITEM4

LOOPAAT

FNODE#	TNODE#	LPOLY#	RPOLY#	LENGTH	LOOP#	LOOP-ID	RT-ID	NEXT-ARC	CUM-DMNO	CUM-IMPED

ARC IDENITY LOOP POLY OVRCOVER LINE

OVERCOVER.AAT

FNODE#	TNODE#	LPOLY#	RPOLY#	LENGTH	OVRCOVER#	OVRCOVER-ID	LOOP#	POLY#

Be sure to name LOOPAAT and POLYPAT back to LOOP.AAT and POLY.PAT

The result of the identity is to create a third normal form output file.