Relational DBMS concepts for the New user

Jack Horton

Why use an RDBMS?

- Data maintenance
- Standardized access
- Multi-user access
- Data protection

RDBMSs offer Data protection

- Recovery
- Concurrency
- Security

Data protection

- Recovery from
  - User error
  - Statement or process failure
  - Instance failure
  - Media failure
- Recovery must be:
  - Predictable – all committed transactions, no uncommitted trans
  - Reliable
- Databases use optional transaction logging to allow recovery to the time just before the point of failure

Data protection

- **Concurrency – simultaneous access to the same data**
  - **Multiple users modifying a table at the same time**
  - **Read consistency**
  - **Transactions are isolated from each other**
  - **Locking**

Data protection

- Security
  - Users are assigned privileges
    - Privileges on individual objects
    - Privileges to perform specific actions
    - Resource usage privileges
  - Auditing
    - Tracking the actions a user performs

RDBMS data storage

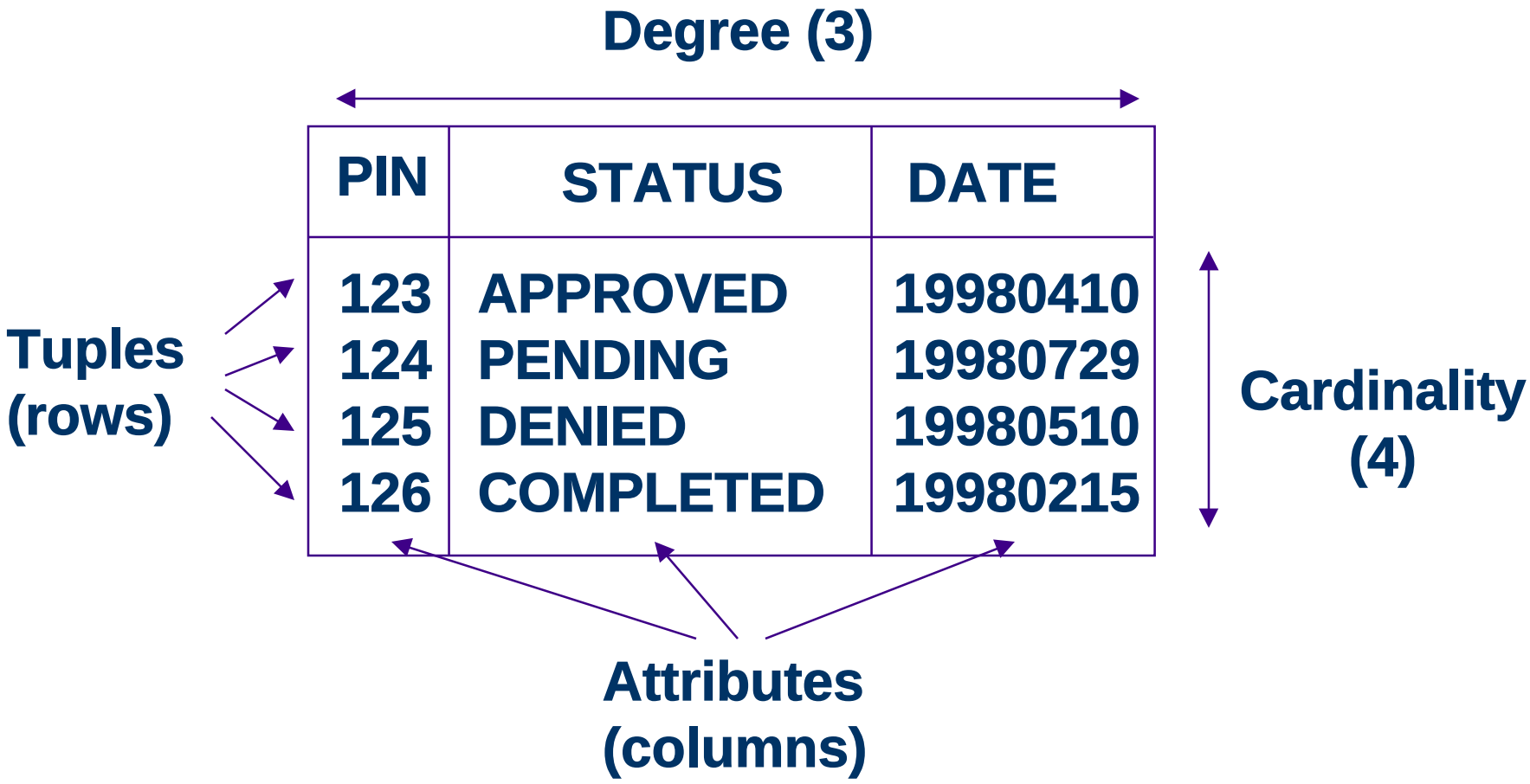
- All RDBMS need a way to obtain exclusive access to disk resources.
- Some systems prefer “raw” filesystems, others prefer them “cooked.”
- Known as “tablespaces,” “devices,” “disks,” “chunks,”

RDBMS organization

- **Catalog or Data Dictionary**
  - **An RDBMS addresses metadata with the same mechanisms as user data.**
  - **The “catalog” of an RDBMS is the set of system objects required to implement itself.**

Relational theory

- **Relation**
  - **The theoretical structure that is implemented as a ‘table’**



Relational theory

- **Relation properties:**
  - **Each tuple is distinct (no duplicates)**
  - **Tuples are unordered (top->bottom)**
  - **Attributes are unordered (left->right)**
  - **All attribute values are atomic (relations do not contain repeating groups)**

Relational theory

- Attribute values are taken from pools of legal values known as “domains.”
- In some cases, it is desirable to mark an attribute value as “empty” or “missing.” This can be achieved through the use of “NULL” values.

Relational theory

- Keys
  - Candidate
    - Provide tuple-level addressing system
    - May be more than one per relation
  - Primary
    - One per relation (others are alternate keys)
  - Foreign
    - Attribute of a relation which refers to primary key in another relation

RDBMS organization

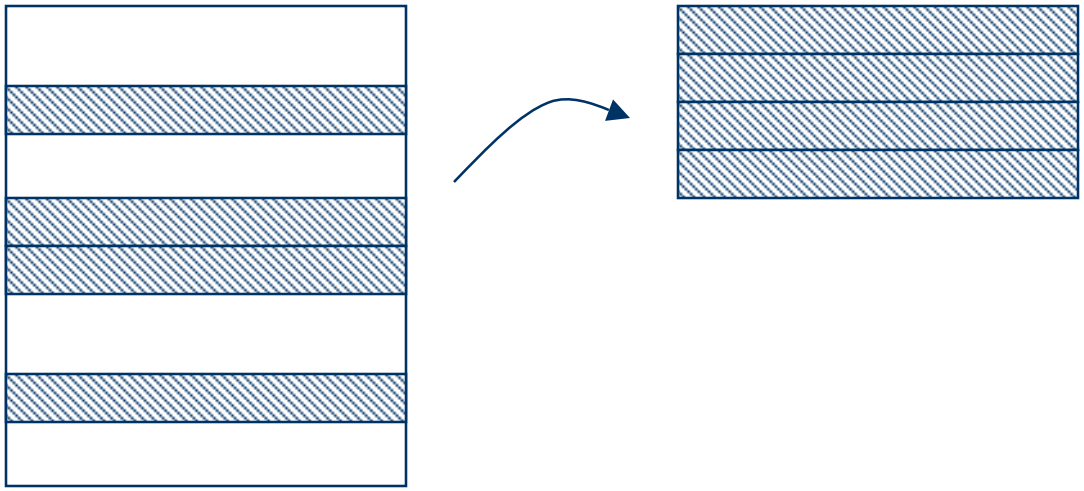
- Objects
  - TABLE
  - VIEW
  - INDEX
  - TRIGGER
  - PROCEDURE

Relational algebra

- Relational theory is a mathematical model. It defines a number of operators and properties for the interaction of relations.
- One key concept is closure -- the output of any relational operation is another relation. This makes relational algebra very powerful!

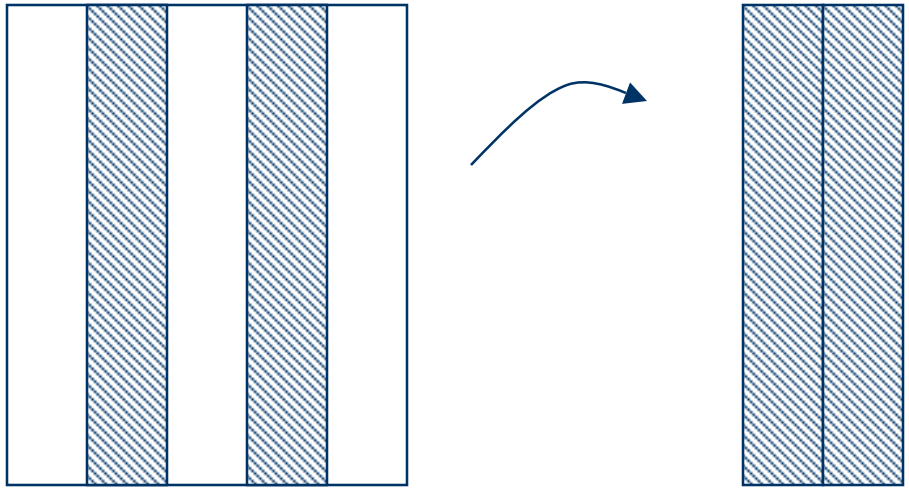
Relational algebra

- **Restrict**
  - **Produce a tuple subset**



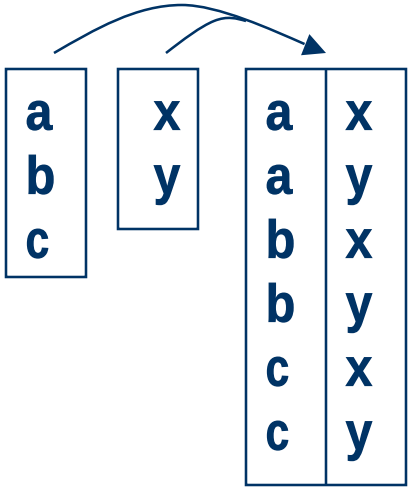
Relational algebra

- **Project**
  - **Produce an attribute subset**



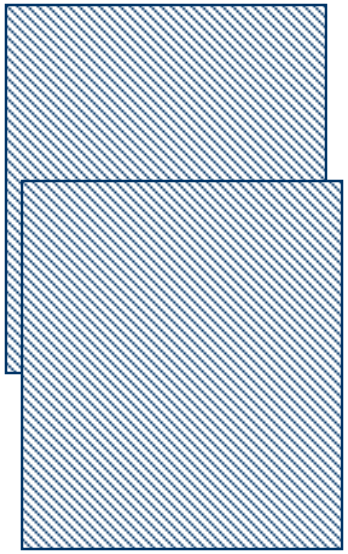
Relational algebra

- **(Cartesian) Product**
  - **Produce all possible combinations**

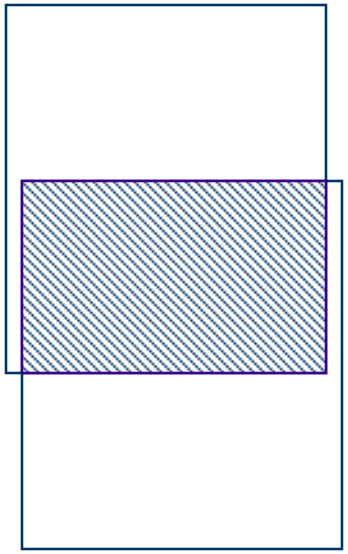


Relational algebra

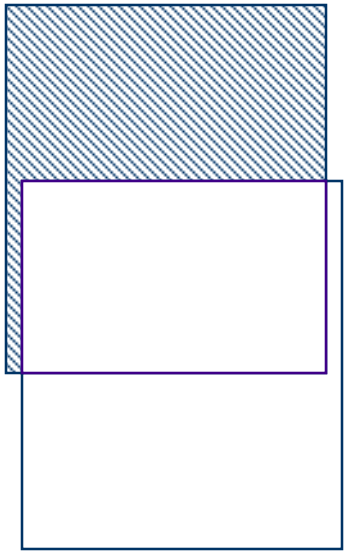
Union



Intersection

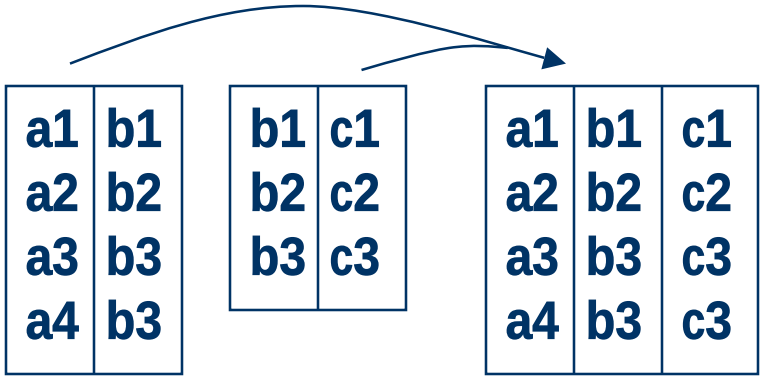


Difference



Relational algebra

- **(Natural) Join**
  - **Produce all possible tuples which are a non-repeating combination of two tuples based on shared attribute value(s).**



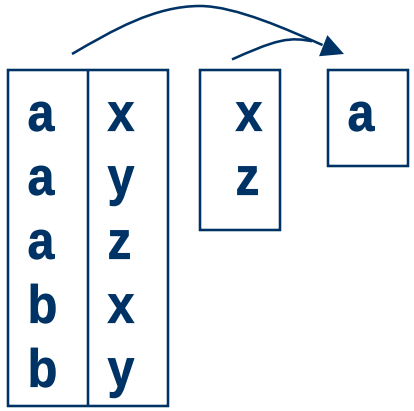
a1	b1
a2	b2
a3	b3
a4	b3

b1	c1
b2	c2
b3	c3

a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b3	c3

Relational algebra

- **Divide**
  - **Given binary and unary relations, produce all values of the non-common attribute which are present for all unary tuples.**



SQL

- Structured query language
- ANSI standard
- Broken into two components
  - Data Definition Language (DDL)
  - Data Manipulation Language (DML)

SQL

- **To create a table use the CREATE TABLE command:**

```
CREATE TABLE table_name ( column_def_list )

CREATE TABLE PARCELS (
        PARCEL_NO          INTEGER,
        ASSESSED_VALUE     FLOAT)
```

- **The DROP command is used to remove objects from catalog:**

```
DROP TABLE PARCELS
```

SQL

- **The most common SQL command is the SELECT statement.**
- **The basic format of a SELECT statement is:**

  SELECT *column_list* FROM *table_list* {WHERE *where_clause*} {ORDER BY *column_list*} {GROUP BY *column_list*}

SQL

- **Examples (I)**

```
SELECT * FROM PARCELS

SELECT PARCEL_NO FROM PARCELS

SELECT PARCEL_NO FROM PARCELS WHERE
ASSESSED_VALUE > 360 AND TAX_DISTRICT = 'MRR'
```

SQL

- **Examples (II)**

```
SELECT      b.OWNER_NAME as “Owner”,
a.ASSESSMENT as “Cost”
FROM        PARCELS a, OWNERS b
WHERE       ASSESSED_VALUE > 360 AND
TAX_DISTRICT = ‘MRR’ AND        a.PARCEL_NO =
b.PARCEL_NO
```

SQL

- **ORDER BY**
  - Applies sort to resulting rows
- **Example**

```
SELECT          b.OWNER_NAME as “Owner”,
a.ASSESSMENT as “Cost”
FROM            PARCELS a, OWNERS b
WHERE           ASSESSED_VALUE > 360 AND
TAX_DISTRICT = ‘MRR’ AND                a.PARCEL_NO =
b.PARCEL_NO
ORDER BY        OWNER_NAME
```

SQL

- **GROUP BY**
  - **Used to summarize one or more rows**
  - ***column_list* must be subset of selected columns**
  - **Used in conjunction with summarization functions (SUM, MIN, MAX, …)**
- **Example**

```
SELECT    a.OWNER_NAME as “Owner”,
SUM(b.ASSESSMENT) as “Total”
FROM      OWNERS a, PARCELS b
WHERE     a.PARCEL_NO = b.PARCEL_NO
GROUP BY          OWNER_NAME
```

SQL

- **"Subselects"**
  - **The closure property allows for substitution of SELECT expressions into SQL statements**
- **Examples**

```
INSERT INTO PARCELS_BAK
    SELECT * FROM PARCELS


SELECT PARCEL_NO FROM PARCELS
    WHERE ASSESSED_VALUE >
              (SELECT AVG(ASSESSED_VALUE)
                        FROM PARCELS)
```

SQL

- **The basic format for the INSERT command is:**

  INSERT *source* INTO *target(columns)* VALUES *(value_list)*

- **Example:**

  INSERT INTO PARCELS (PARCEL_NO,ASSESSED_VALUE)
  VALUES (125,180)

SQL

- **The basic format for the UPDATE command is:**

  UPDATE *target* SET *col_val_pair_list* {WHERE *where_clause}*

- **Example**

```
UPDATE PARCELS
   SET ASSESSED_VALUE = 200
   WHERE PARCEL_NO = 123
```

SQL

- **The basic format for the DELETE command is:**

```
DELETE FROM table {WHERE where_clause}
```

- **Example:**

```
DELETE FROM PARCELS
            WHERE PARCEL_NO = 121
```

Indexing

- Queries represent the majority of operations performed on tables.
- Exhaustive searching of rows is SLOW!
- Indexes are used to store a sorted list of key values, allowing for faster searches.
- Examples
  - CREATE INDEX PARCELS_PN
    ON PARCELS(PARCEL_NO)
  - CREATE INDEX PARCELS_AV
    ON PARCELS(ASSESED_VALUE,
    PARCEL_NO)

Performance issues

- **Many strategies have evolved in order to maximize RDBMS query performance:**
  - **Block I/O**
    - **Since locating the starting byte of a data page is the slowest part of a I/O request, accessing data in large "blocks" makes sense.**
  - **Caching**
    - **Disk access is very slow in comparison to memory access (~10,000x).**
    - **A "cache" is a copy of most recently read pages.**
    - **A "blown cache" penalty can occur if the cache is smaller than the frequently-read pages.**
  - **Indexing and Query optimization**
    - **In a complex query, the order of the search is important**
    - **Example: it is faster to first search on age, then on sex, to query for the 25 year old males**
    - **Rule-based optimization; cost-based optimization; hints**

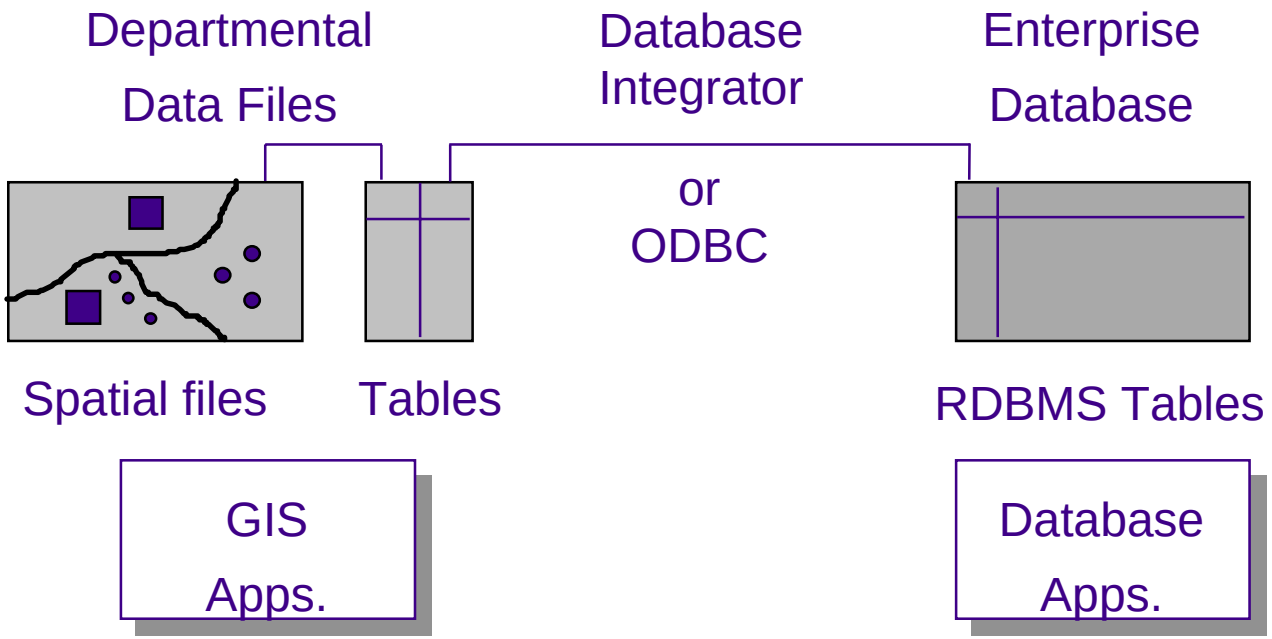
SDE stores GIS data in an RDBMS

- GIS users want better data management
  - data integrity
  - fast access for many simultaneous users
  - efficient use of the network
  - common environment to manage spatial and tabular data
  - SQL standard
- MIS users want spatial functionality
  - include spatial data as a managed enterprise asset
  - support GIS applications
  - spatially enable applications.

    Example:
    - Point-in-polygon query to determine auto insurance rate.
    - Operator types in address, rate appears on screen.
    - The operator never sees a map.

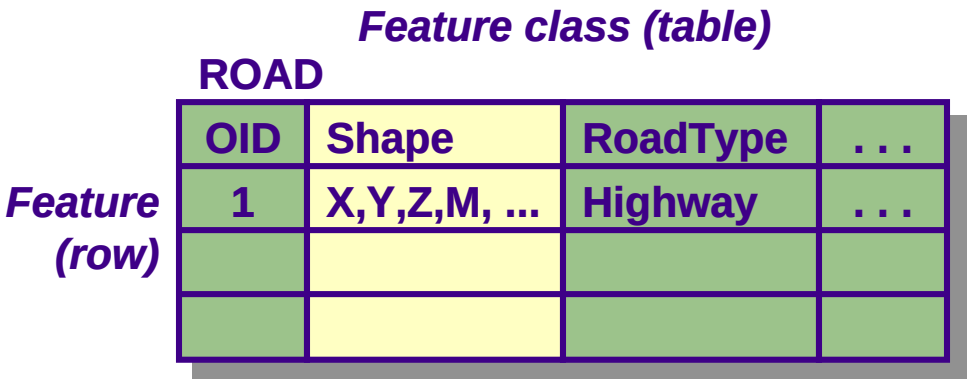
Traditional GIS data is not stored in RDBMS

- **Problems coordinating transactions on data**
- **No referential integrity**
- **Different application environments**



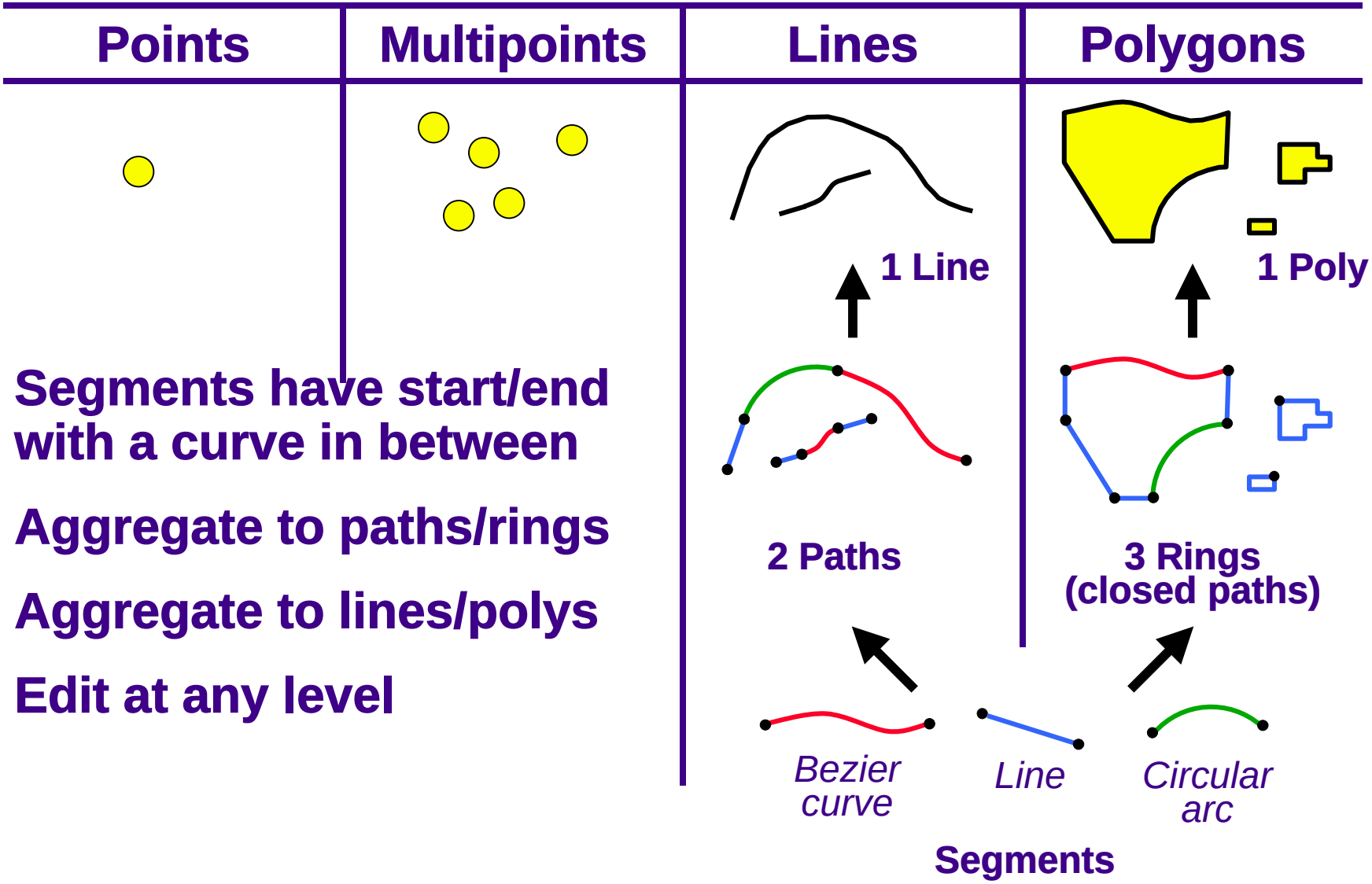
Features

- Features are spatial objects
  - object with a geometry attribute
  - Vector model for geographic entities
  - Features (rows) belong to feature classes (tables)



- Feature location can be stored three ways:
  - Binary
  - Normalized
  - Extended Type

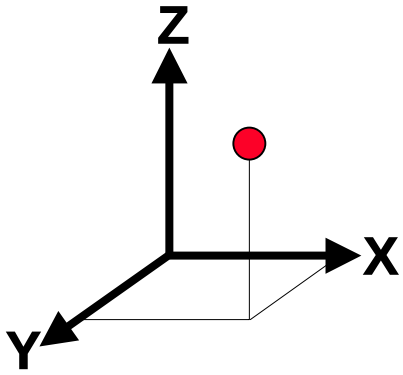
Feature geometry

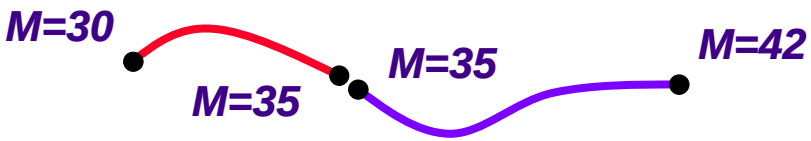


- Segments have start/end with a curve in between
- Aggregate to paths/rings
- Aggregate to lines/polys
- Edit at any level

Feature coordinates

- **Feature coordinates are:**
  - **X - position in X**
  - **Y - position in Y**
  - **Z - position in Z (optional)**
    - **E.G.; elevation for a point or line segment end**
  - **M - a measurement (optional)**
    - **E.G.; Dynamic Segmentation measures**



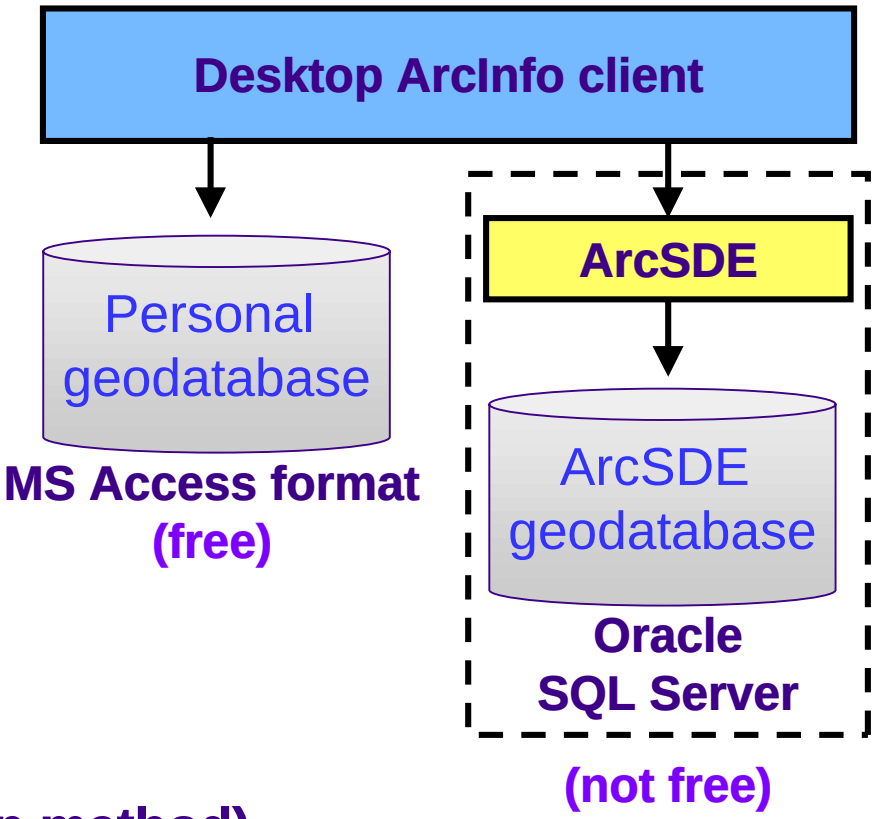


One line made of two segments

- **Stored as integer (scaled in Spatial Reference)**

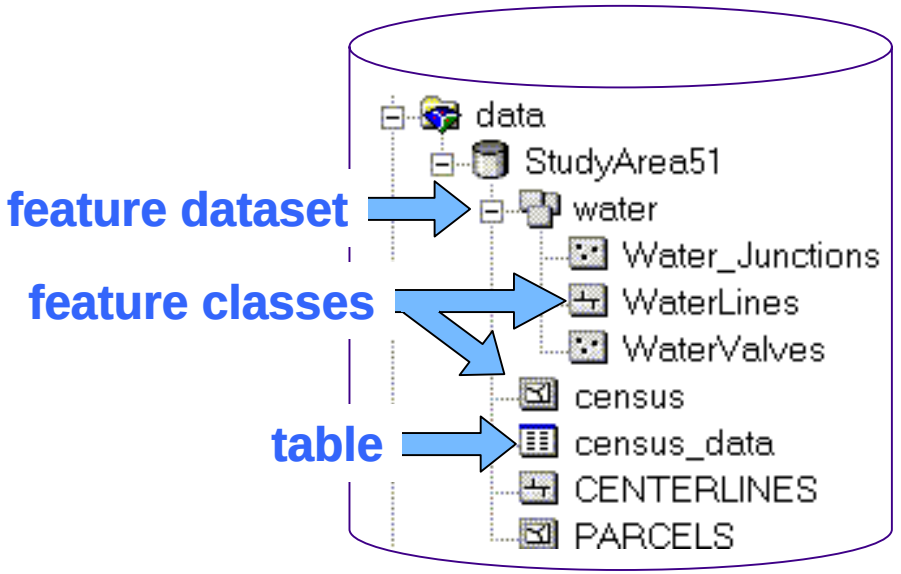
Storing the geodatabase

- **Personal geodatabase**
  - Stored in an .mdb file
  - Automatically connected through JetEngine server
- **ArcSDE geodatabase**
  - Stored in an RDBMS
  - User connects through ArcSDE server
- **The difference**
  - Type of RDBMS (and connection method)
  - Multi-user editing and conflict resolution tools (ArcSDE)
- **Once loaded, use same tools on either storage type**



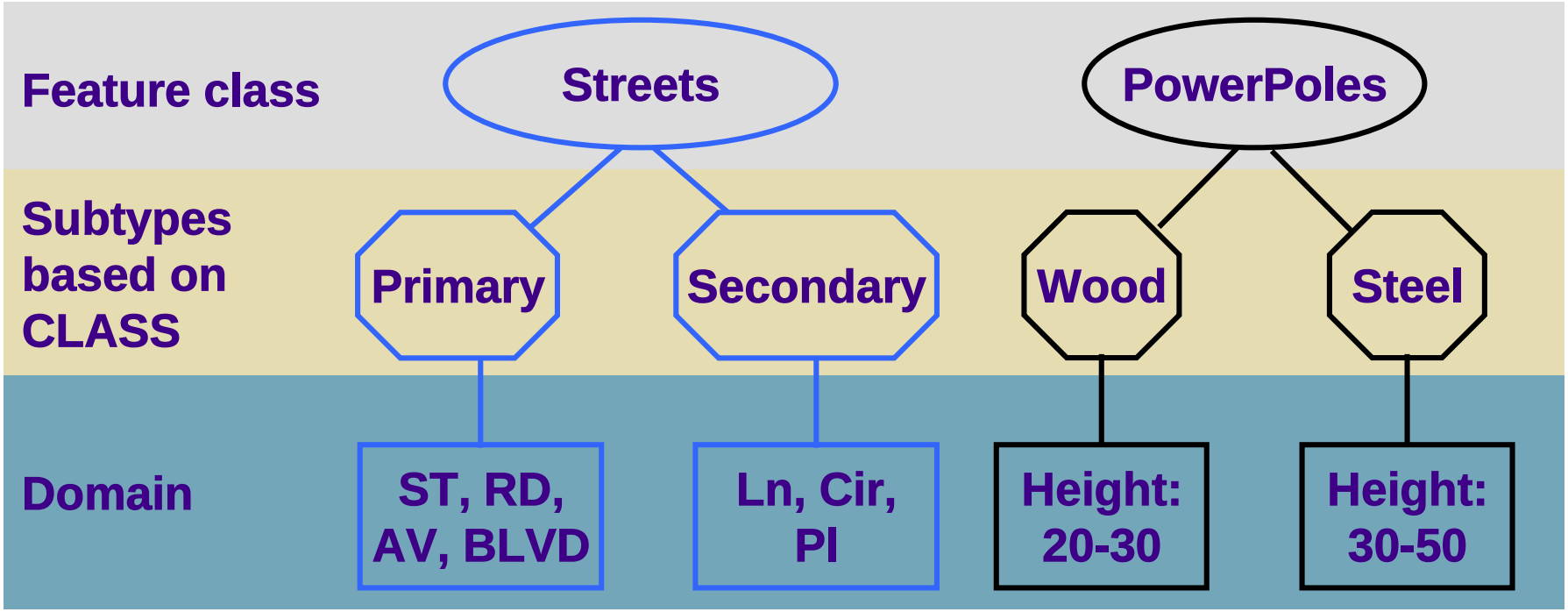
Geodatabase basics

- Stores tables, feature classes, feature datasets, more …
- Tables
  - A collection of attribute rows and columns
- Feature classes
  - A collection of features
  - Conceptually like a shapefile
- Feature datasets
  - A collection of feature classes
  - Conceptually like a coverage
- Raster datasets
- Rules
  - Domains
  - Connectivity rules



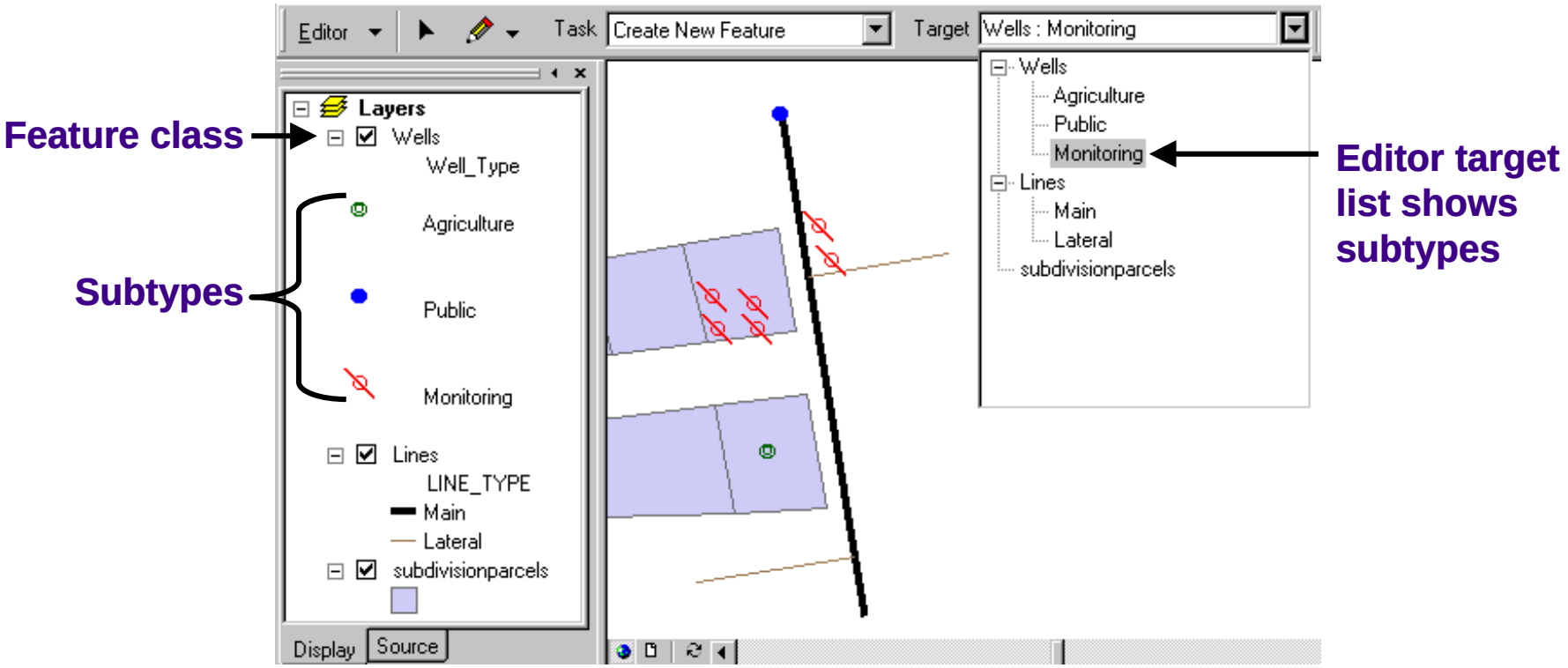
Introducing subtypes and domains

- **Prevent illegal attribute assignment to features, tables**
  - **Subtype - a subset of records within a field**
  - **Domain - a definition of valid values for a field or subtype**



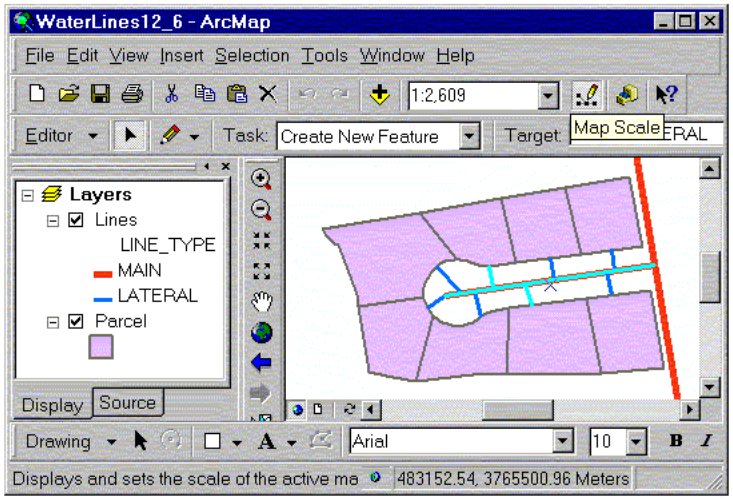
Subtypes in ArcMap

- Add, edit, symbolize by subtype
  - New features get subtype defaults for attributes

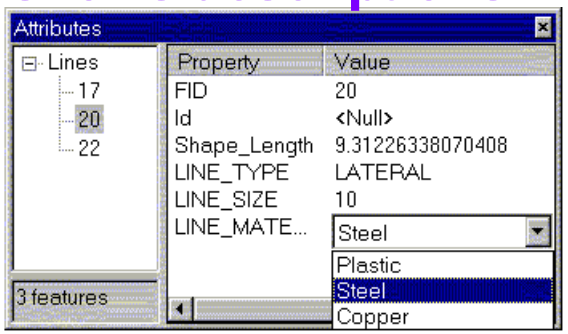


Editing records with coded value domains

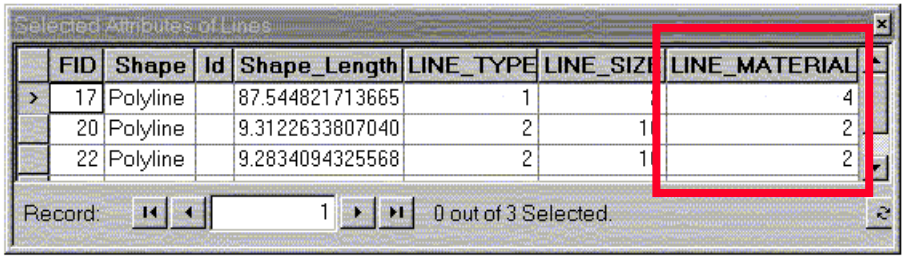
- Attribute editor only shows valid values
- The description is displayed instead of the code



Attribute editor shows descriptions...

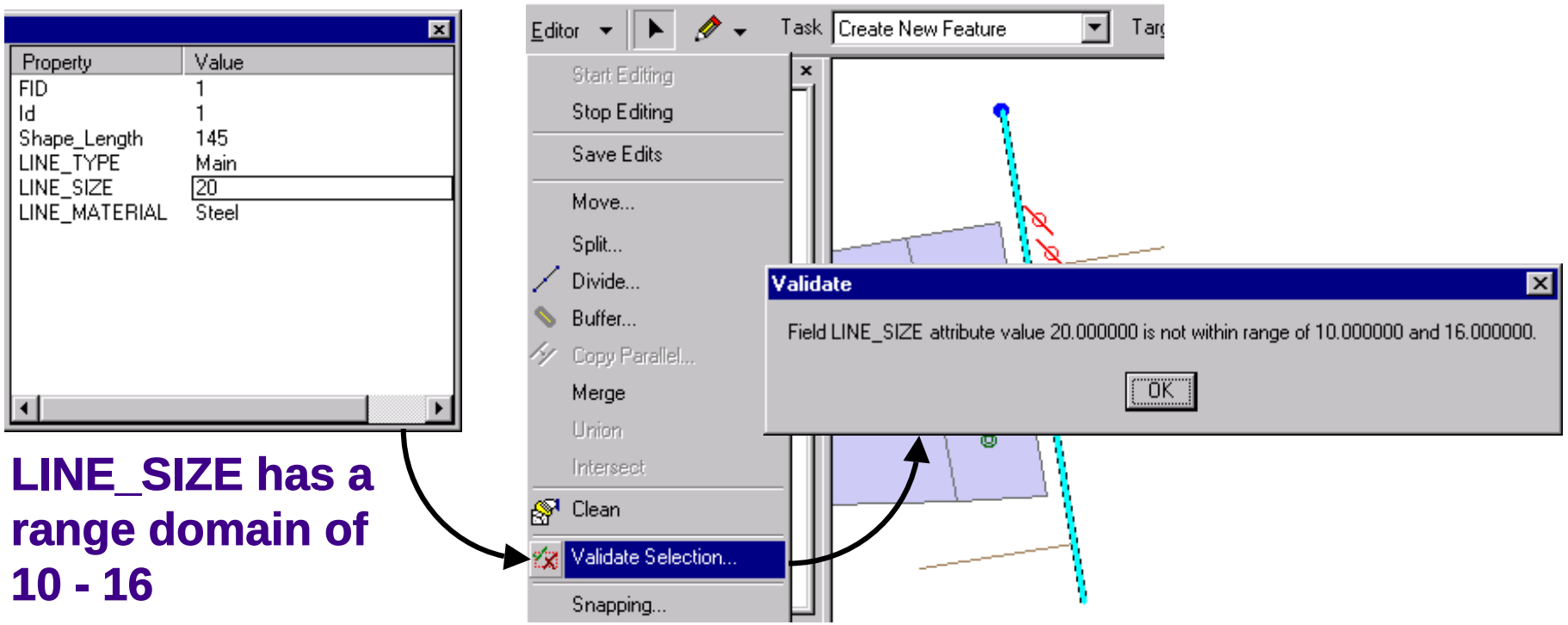


... but the underlying table stores the codes



Editing records that have range domains

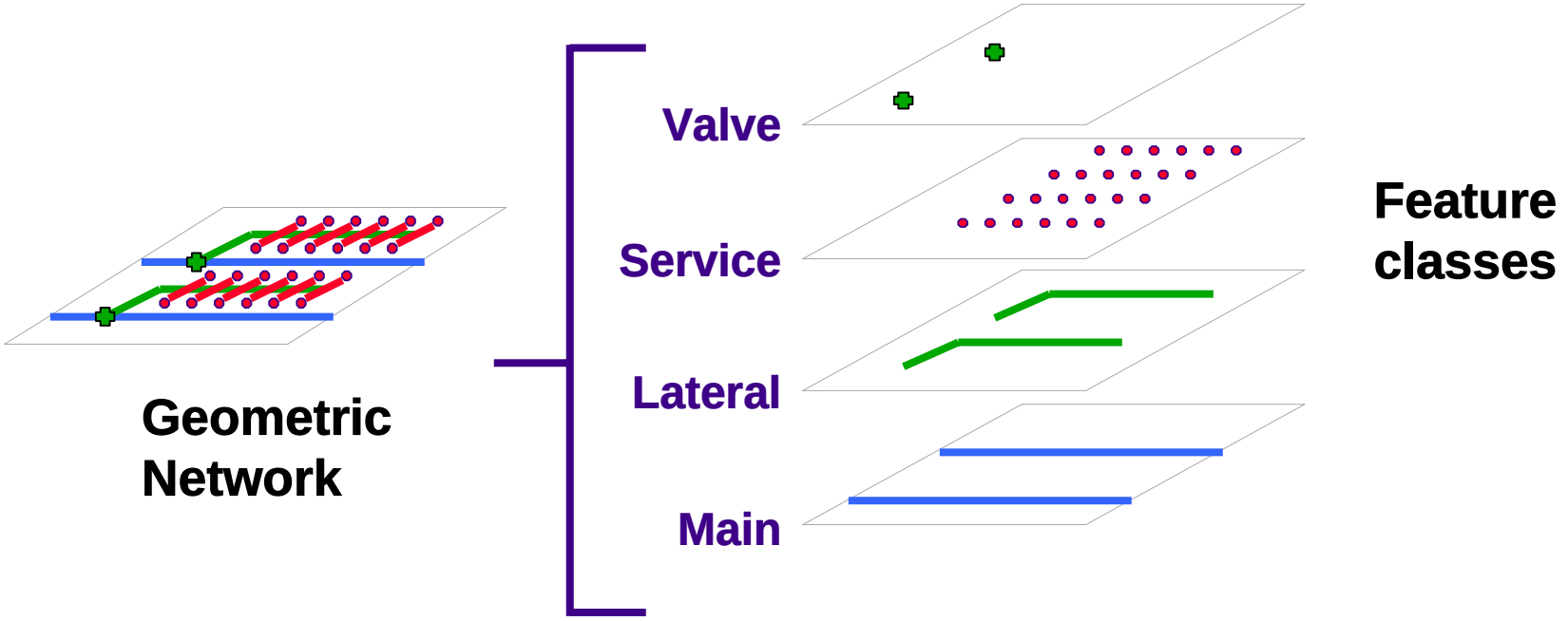
- Perform edit in ArcMap
- Use *Validate Selection* to verify edit against range
  - Invalid features remain selected



LINE_SIZE has a range domain of 10 - 16

Introducing a Geometric network

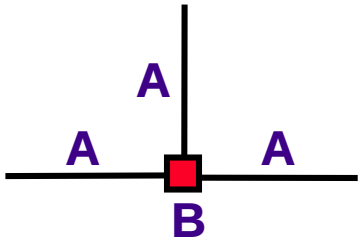
- Feature classes in a single feature dataset
  - A feature class can only participate in one network
- Connectivity based on geometric coincidence



Connectivity rules

- Validate using validate selection
- Edge - Junction
  - Cardinality
    - Number of junctions connecting to an edge
    - Number of edges connecting to a junction

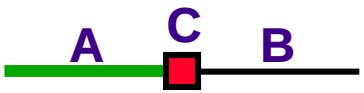
Junction “B” can connect to 3 “A” edges

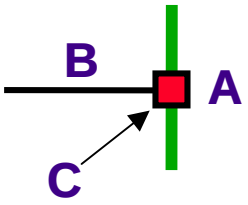


Edge “A” can have up to 2 “B” junctions attached

- Edge - Edge
  - Default junction is automatically added during editing

Edge “A” can connect to Edge “B” through junction “C”

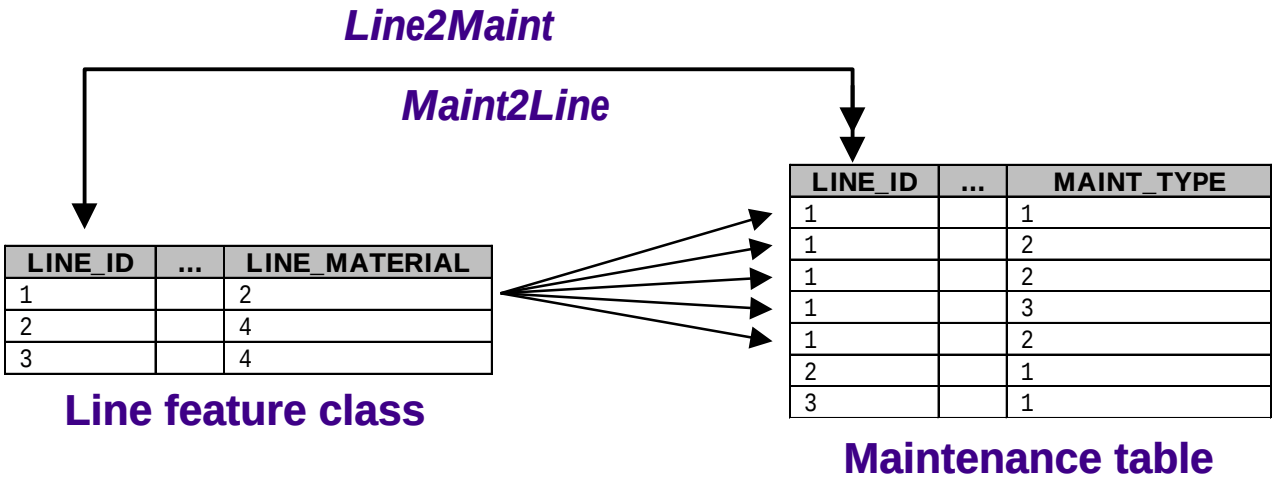




Adding “B”
Snap to “A”
“C” is automatically created

Relationships

- An association between tables or feature classes



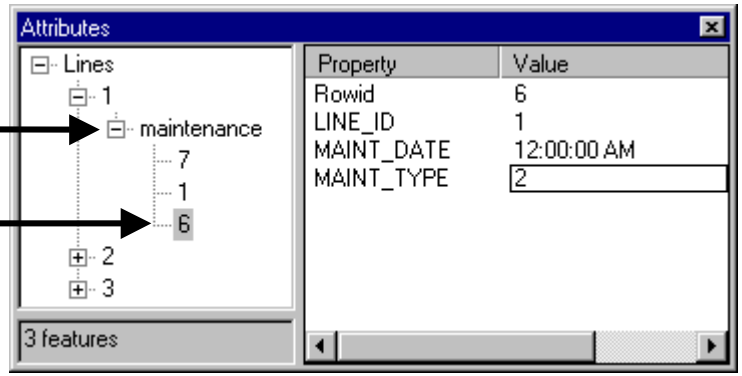
- Works with the geodatabase and coverages
- Tables must be in the same workspace
  - Same data type

Accessing related records in ArcMap

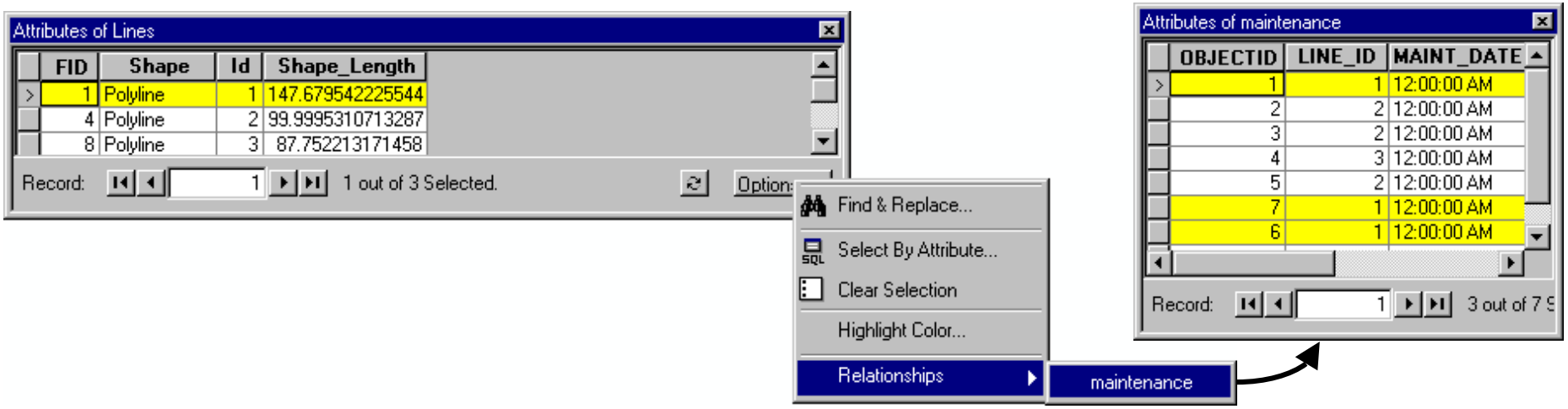
- Fields in related table appear in Attributes editor

Edit all related records

Edit single related records



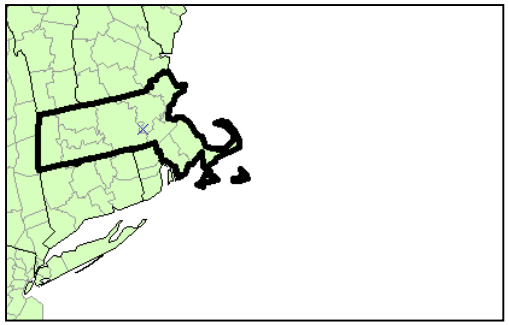
- Open related table
  - Selected records in related table update when table is displayed



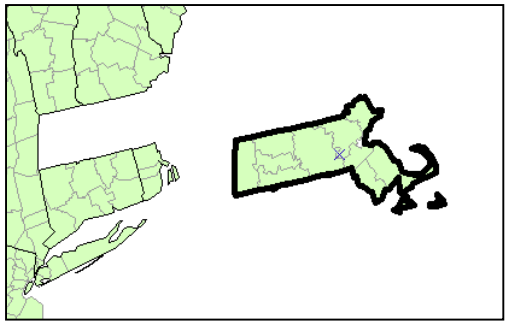
Simple and composite relationships

- **Simple**
  - **Objects exist independently**
- **Composite**
  - **Destination objects cannot exist without origin objects (deleted)**
  - **Destination features move with origin features**

Composite relationship, State to County



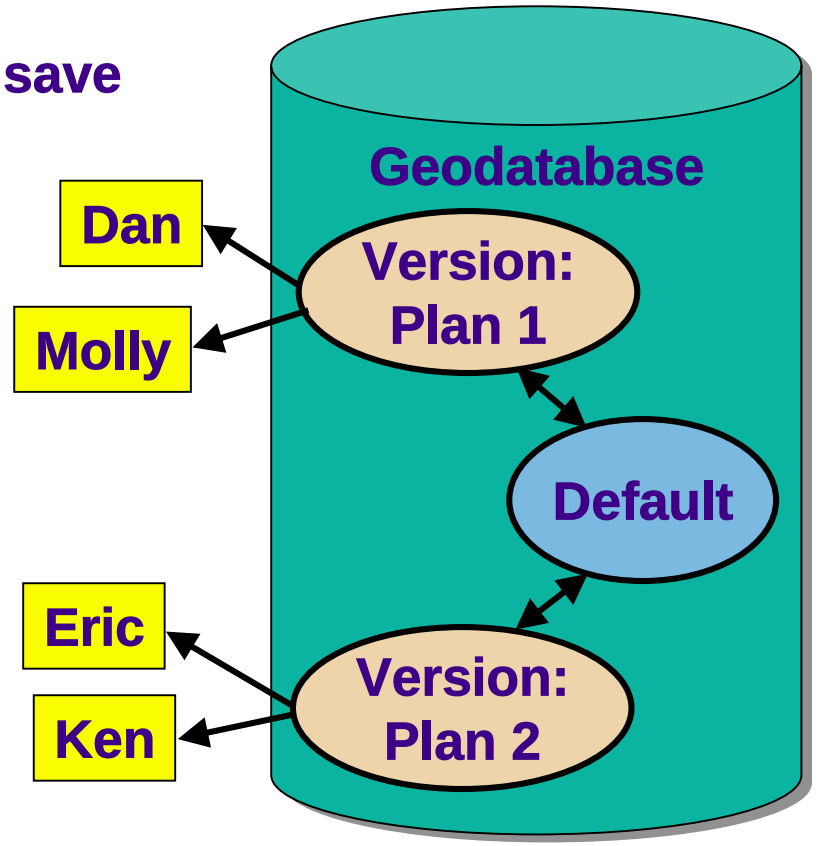
Select state and move it…



the counties follow

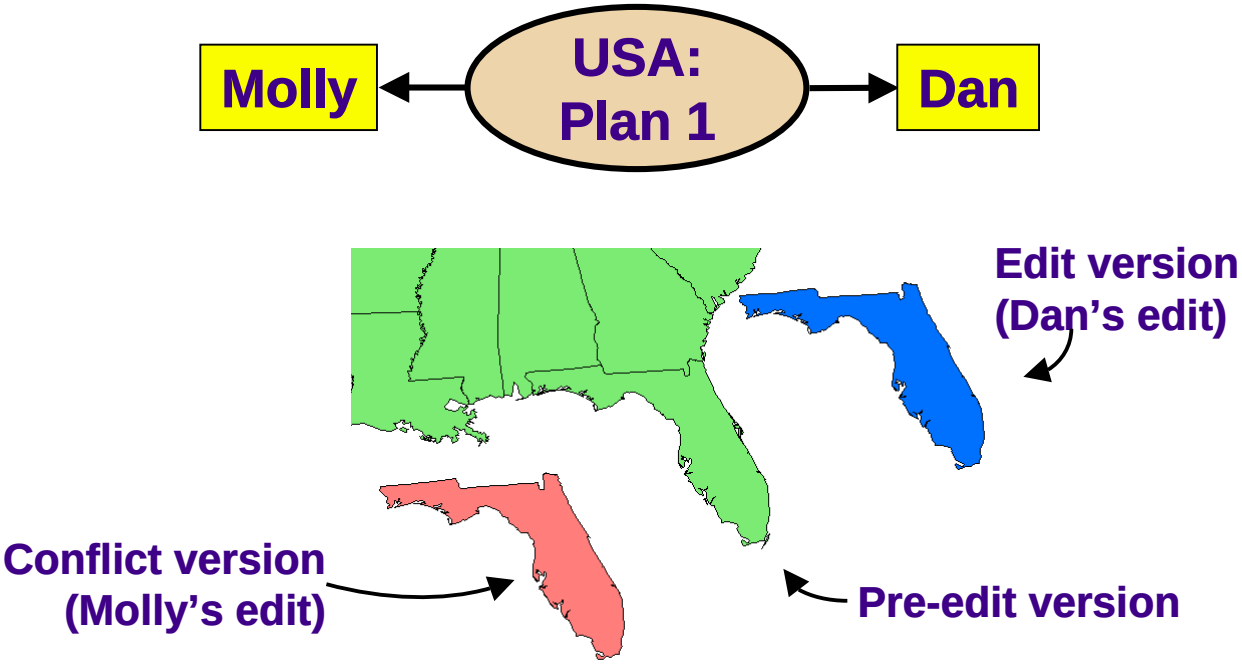
Multi-user editing

- Many users may edit any version simultaneously
  - You see your edits only
  - Others see your edits when they save
- Version permissions
  - Private
    - Owner views and edits
  - Public
    - Everyone views and edits
  - Protected
    - Everyone views; owner edits



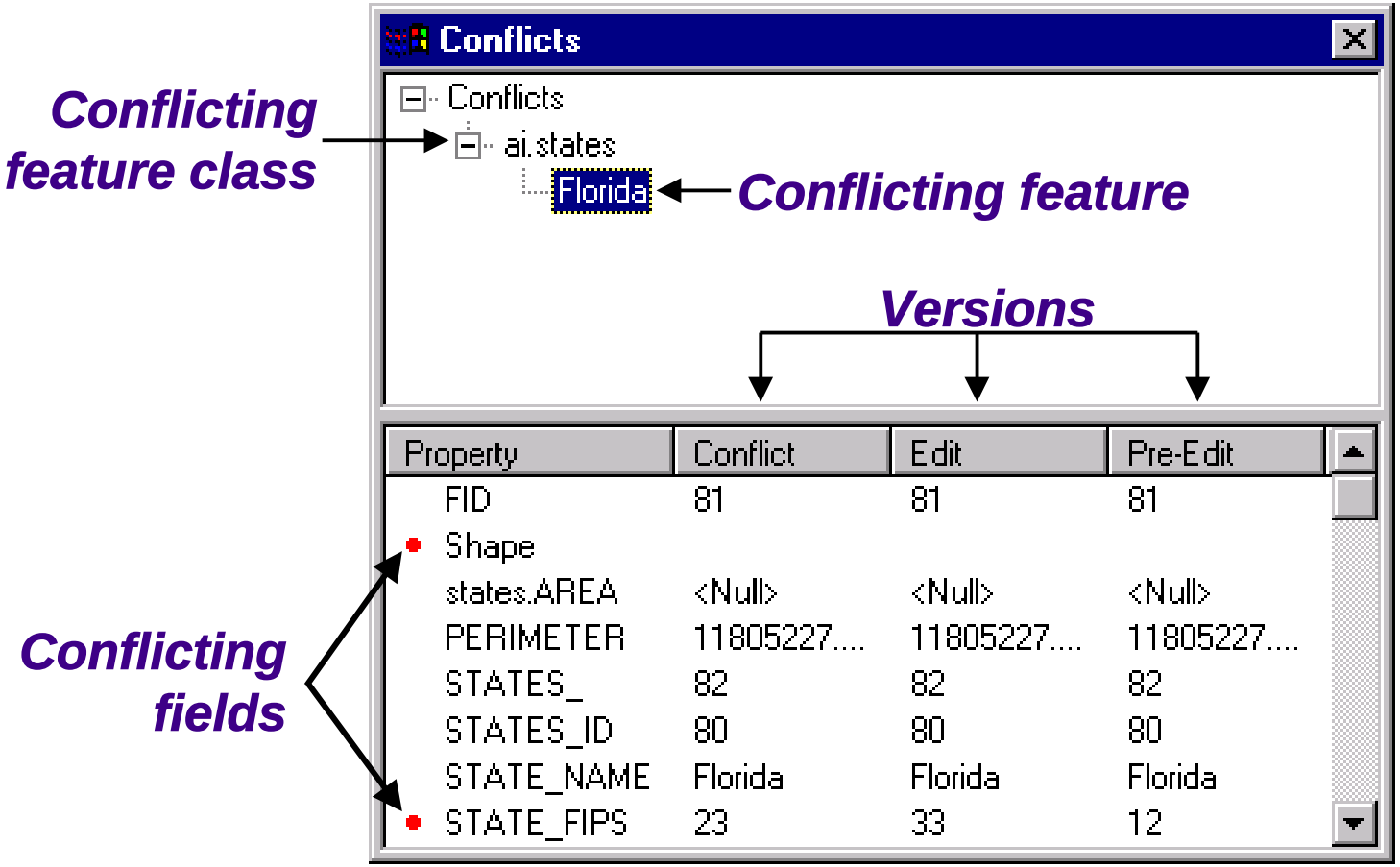
Conflicts

- Occur when two users edit the same feature
  - A coordinate and attribute edit can create a conflict
- The second person to save will notice the conflict



Displaying conflicts

- **Triggered by Save or Reconcile**
  - **Dan tried to save after Molly edited the same feature**



Summary

- RDBMS principles
- SQL
- Performance
- SDE
- Geodatabase


geography
gis
esri
our global network
ESRI