CORPORATE GIS STANDARDS
DATABASE

TABLE OF CONTENTS

CORPORATE GIS STANDARDS
DATABASE

I. Introduction

A. Purpose

Compliance to these standards will facilitate data sharing, integration and compatibility within the GIS System. The objective of this standard is to provide guidelines for database development within the City and County of Denver and its contractors.

B. Reference

It is the responsibility of the user to ensure they are using the most recent version of the standards. The most recent version for all Corporate GIS Standards can be found at http://mydenver/, under DenverGIS, then under Standards.
Inquiries should be directed to:

David Luhan
GIS Supervisor
City and County of Denver
(720) 865-2670
DenverGIS@ci.denver.co.us

II. Overview

Although a database can simply be defined as a collection of data, it is important to note that this document refers specifically to attribute databases – databases that do not contain geometry (points, lines and polygons). This includes traditional relational databases (Oracle, SQL Server, Access) and non-relational tabular formats (text files, dBase files, Excel spreadsheets).

Note: For datasets that do contain geometry (ex. ArcInfo Coverages, Shapefiles, AutoCAD drawings, ArcInfo Export or Generate files etc.) please refer to Spatial Data Standards.

III. Spatial Reference

The following spatial reference standards enable non-spatial databases to be referenced to spatial locations. This implies that the records in the database must either relate to geometric features (links to points, lines or polygons) or that records in the database can be used to determine spatial location and establish geometry (location is implied, for example by address or GPS coordinate). This document will refer to the former as 'key-based' spatial reference and to the latter as 'implied' spatial reference.

A. Key-based Spatial Reference

CORPORATE GIS STANDARDS
DATABASE

Key-based spatial referencing occurs by establishing a common key between a database and a spatial dataset. A key is a common column in both the spatial and attribute database that is used to establish a relationship between the records. Records from the database are then related to geometric features by simply matching the key for each record in the database.

Example: The spatial dataset parcels (defines parcel geometry) contains a column (key) identified as SCHEDNUM, the parcel schedule number. An attribute database to be related to the parcels layer by schedule number must contain the same column (key). The key from each dataset is used to spatially reference the database.

The following standards must be adhered to use key-based referencing.

1. Datatype

The datatype chosen for the database key must match the datatype of the key in the spatial dataset that the data is to be related to (Please contact Corporate GIS to ensure that the key to be used is appropriate). The datatype defines the type of data that is stored in the key column and is typically string ('ABCDE') or numeric (12345) datatype. The specific datatype chosen is dependant on the database being used. Refer to **Database Formats** to determine the appropriate datatype to choose when defining the database key.

Note: The key field should be a code or numeric value. This increases join performance and avoid errors that could result from mistyping longer keys (ex. full names).

2. Column Length

The length of the key column must match the length of the related column in the spatial dataset. This ensures that values in both columns can be matched and constrains the length of the data that can be entered to avoid potential mismatches.

3. Unique Key Constraint

The key column in the database *must* contain unique values (no duplicates) to enable a relationship to be established to a spatial dataset. Only the following relationships can be established, each of which require that the database key be unique:

One-to-one: Each record in the database relates to exactly one record in the spatial dataset.

One-to-many: Each record in the database relates to one or more records in the spatial dataset.

In the supported RDBMS (Access, SQL Server, Oracle) it is possible to define a unique constraint on a column to ensure that no duplicates can be entered. It is highly recommended that this constraint be enabled to ensure data integrity.

4. Field Format

To ensure that the values in the key column match the values in the spatial database the field format must be matched:

Example: The SCHEDNUM column in the parcels spatial dataset is defined as a 13 digit character string. The format is as follows:
MAPNUMBER(5 chars)
BLOCKNUMBER(2 chars)
PARCELNUMBER(3 chars)
APPENDAGE(3 chars).
Example: 0313403012000
Each section is left padded with zeros (0) as required (ex. 03134). This format must be followed exactly to ensure matching.

If supported by the database, the field format should be validated on data entry to ensure data integrity.

5. Flow Direction (If applicable)

In some cases it may be necessary to reference data to linear features that represent flow.

Example: Traffic counts on a street segment may have an associated direction. An additional column must be included in the database to enable the flow direction to be addressed when spatially referencing the data.

The following recommendation is provided:

a) Database Column

The following column should be present in the database table:

FLOW_DIR

Description:	Attribute indicating flow/travel direction along the segment that the key relates to.
Example:	1
Data type:	Number
Width:	1
Valid entries:	0 – No flow direction or flow in both directions 1 – Flow in direction of the segment 2 – Flow opposite to direction of segment

Note: Flow information should only be linked to datasets whose features have been intentionally captured to represent directionality. Please contact Corporate GIS if you are intending to link to features and represent flow. Examples of datasets that include flow information are streams (features captured in downstream direction), sewer lines (features captured in flow direction), and streets (features captured in direction of increasing address range).

B. Implied Spatial Reference

An alternative to establishing links to geometric features is to include information in the database that can be used to create geometry. This generally applies to point locations only.

Example: A database contains addresses and associated values for each address. The addresses imply a spatial location and can be used to generate point features that represent those locations. The generated point features are then used to reference the data spatially in the Corporate GIS Spatial Database.

The following standard methods are provided.

1. Street Address Geocoding

Geocoding is the process of determining location by physical (street) address (non-physical addresses such as PO boxes cannot be geocoded!). Databases that contain standardized addresses can be spatially referenced by creating point features to represent the address locations.

Denver Corporate GIS adheres to the Delivery Address Line formatting standard for physical addresses as specified in Section 23 of the US Postal Service Postal Address Standards, Publication 28, November 2000 (http://pe.usps.gov/cpim/ftp/pubs/Pub28/pub28.pdf). These standards

must be followed to allow geocoding against the Denver Street Centerline database.

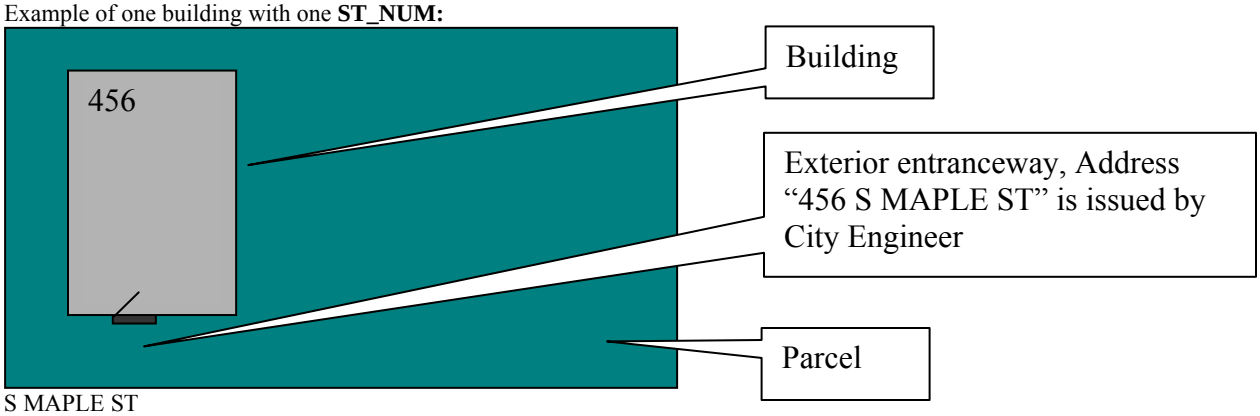
To permit geocoding of addresses the following requirements must be met.

a) Database Columns

The following columns must be present in the database table:

ST_NUM
Description: Address number, issued by City Engineer.
Example: 456
Data type: Number (Long Integer)
Width: 10
Valid entries: May only contain numeric characters [0-9].
Req'd Value: Yes

Example of one building with one **ST_NUM:**



S MAPLE ST

Complete address is: 456 S MAPLE ST

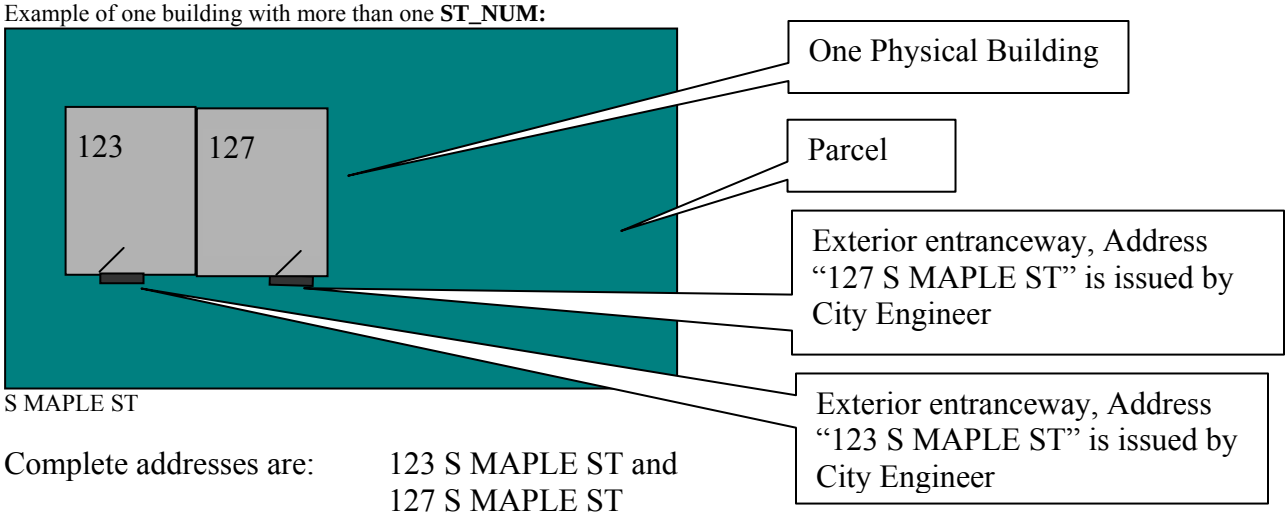
Stored in database as:

ST_N	ST_SUF	ST_PRE	ST_NA	ST_TY	ST_BLDGN	
45		S	MAPL	ST		

The following situation may occur for Town Homes.

Description: Two Address numbers on one building.
Example: 123, 127

Example of one building with more than one **ST_NUM:**



Complete addresses are: 123 S MAPLE ST and
127 S MAPLE ST

Stored in database as two records:

ST_N	ST_SUF	ST_PRE	ST_NA	ST_TY	ST_BLDGN	
12		S	MAPL	ST		
12		S	MAPL	ST		

- **ST_SUFFIX**

  Description: Address number suffix used for legacy addresses only. This attribute is no longer used by the City and County of Denver when assigning new addresses.
  Example: ½, REAR
  Data type: String
  Width: 6
  Valid entries: ½, REAR, UPPER.
  Req'd Value: Yes, if applicable, Not required for Geocoding GIS Street Centerline, Not used for new addresses.

  ST_ PREFIX
  Description: Street pre-directional.
  Example: W
  Data type: String
  Width: 1
  Valid entries: Directional indicator in upper-case.
  Restricted to N, E, S, W. Do not include punctuation.
  Req'd Value: Yes, if applicable

  ST_ NAME
  Description: Primary street name.
  Example: COLFAX
  Data type: String
  Width: 30
  Valid entries: Upper-case street name without punctuation.
  Include only alphabetic characters and spaces.
  Req'd Value: Yes

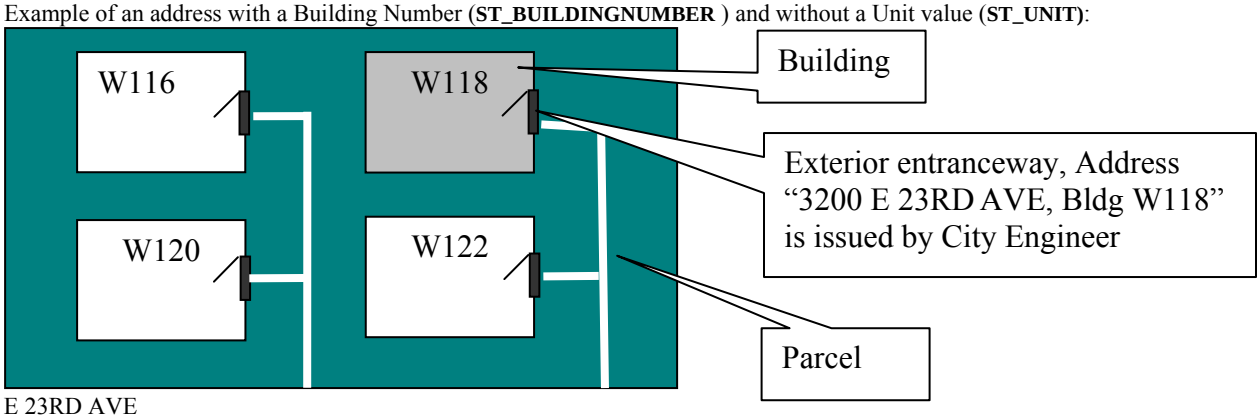
  ST_ TYPE

Description: Street type.
Example: AVE
Data type: String
Width: 4
Valid entries: AVE, BLVD, CIR, CT, DR, HWY, LN, PKWY, PL, RD, ST, WAY
Refer to USPS Postal Addressing Standards for additional values used outside the City and County of Denver.
See Appendix C1 for complete listing.
(Note: The USPS Standard refers to this as the Suffix).
(Note: In Denver, some Street types have been incorporated into the primary street name, refer to list of valid street names).
Req'd Value: Yes, if applicable

- **ST_BLDGNUM**

Description: Building designator.
Example: W118
Data type: String
Width: 6
Valid entries: Varies, usually an alpha value or an alpha value combined with a numeric value, Note that this is a
building number as opposed to a building name.
Req'd Value: Yes, if applicable, Not required for Geocoding GIS Street Centerline

Example of an address with a Building Number (**ST_BUILDINGNUMBER**) and without a Unit value (**ST_UNIT**):



E 23RD AVE

Complete address is: 3200 E 23RD AVE, Bldg W118

Note that string "Bldg" is not stored in the database.

Stored in database as:

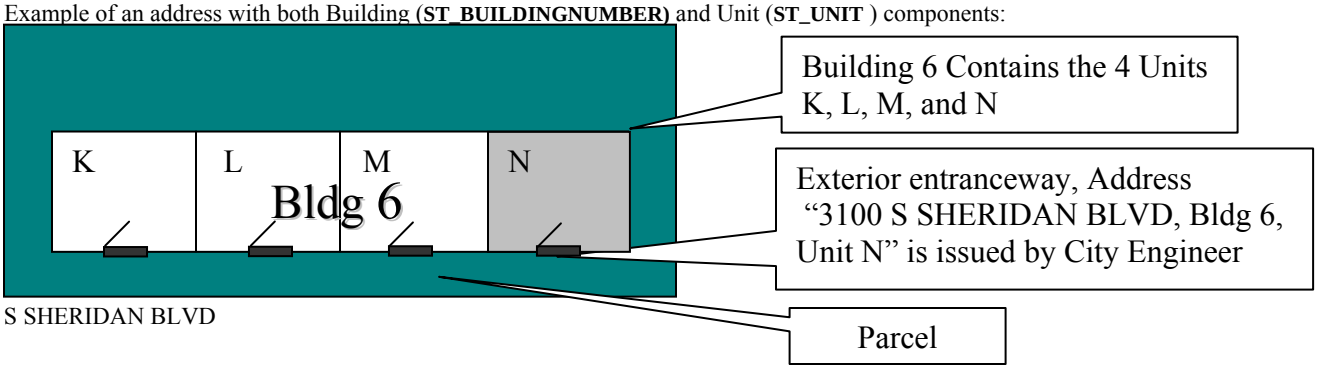
ST_N	ST_SUF	ST_PRE	ST_NA	ST_TY	ST_BLDGN
320		E	23RD	AVE	W116
320		E	23RD	AVE	W118
320		E	23RD	AVE	W120
320		E	23RD	AVE	W121

▲ **ST_UNIT**

Description: Unit Designator used for strip malls or other structures with exterior entrances whose addresses are issued by the City Engineer. The **ST_UNIT** is to be populated only for units with exterior entranceways. Note that these exterior entrances may be a condominium with an exterior entranceway.

Example: K
Data type: String
Width: 10
Valid entries: Varies, usually an alpha value or an alpha value combined with a numeric value
Req'd Value: Yes, if applicable, Not required for Geocoding with GIS Street Centerline

Example of an address with both Building (**ST_BUILDINGNUMBER)** and Unit (**ST_UNIT**) components:



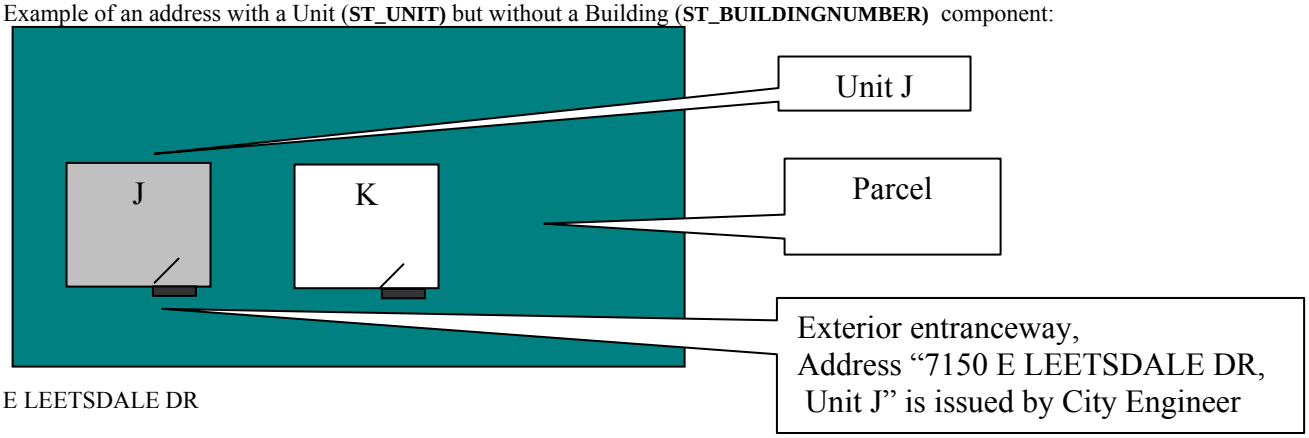
Complete address is: 3100 S SHERIDAN BLVD, Bldg 6, Unit N

Note that neither string "Bldg" nor "Unit" is stored in the database.

Stored in Database as:

ST_N	ST_SUF	ST_PRE	ST_NAM	ST_TY	ST_BLDGN
310		S	SHERID	BLVD	6
310		S	SHERID	BLVD	6
310		S	SHERID	BLVD	6
310		S	SHERID	BLVD	6

Example of an address with a Unit (**ST_UNIT)** but without a Building (**ST_BUILDINGNUMBER)** component:



Complete address is: 7150 E LEETSDALE DR, Unit J

This is an example of an address with a Unit designator but without a Building Number. The Unit value is placed in field ST_UNIT as opposed to ST_SUBADDR because these Units have exterior entrances and are therefore addressed by the City Engineer.

Note that the string "Unit" is not stored in the database.
Note that the configuration shown above represents a legacy address. All newly issued addresses for locations with separate buildings would be populated with two different ST_BLDGNUM values, as opposed to the two ST_UNIT values.

Stored in database as:

T_NU	ST_ FFIX	T_PREF	T_NAM	T_TY	ST LDGNUM	T_UNI	ST_S ADDR	S SUBTYPE
15(			EETSD LE	R				
15(			EETSD LE	R				

SUB-ADDRESSES

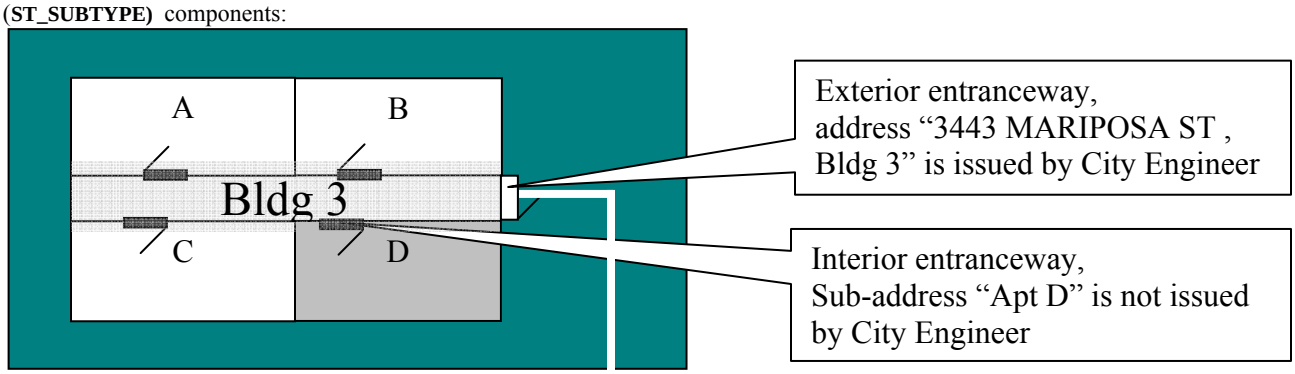
Sub-addresses are defined as the two items (**ST_SUBADDR** and **ST_SUBTYPE**) in an address string that are not assigned by the City Engineer. These represent addresses with interior entranceways.

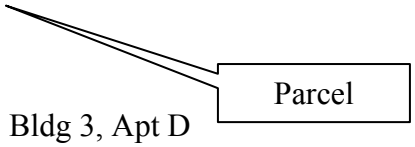
ST_SUBADDR
Description: Sub-address Designator used for condominiums, apartments, suites, or other interior entrances with addresses. The address portion of the complete address is issued by the City Engineer but the Sub-Address portion is issued by the owner. Note that ST_BUILDINGNUMBER may or may not need to be populated.
Example: D
Data type: String
Width: 10
Valid entries: Varies, usually an alpha value or an alpha value combined with a numeric value
Req'd Value: Yes, if applicable, Not required for Geocoding GIS Street Centerline

ST_SUBTYPE

Description: Sub-address Type Designator, Assigned by owner. There can only be a ST_SUBTYPE if there is a corresponding ST_SUBADDR.
Example: Apt
Data type: String
Width: 10
Valid entries: Apt, Suite, and Unit (additional values may be added as required). Note: it is anticipated that values entered into this field will be obtained from a pull down list. The values will be obtained from table **SUBADDRESS_TYPE**.
Req'd Value: Yes, if applicable, Not required for Geocoding GIS Street Centerline

Example of an Address with Building (**ST_BLDGNUM)** , Sub-address (**ST_SUBADDR**), and Sub-Address Type (**ST_SUBTYPE)** components:





Complete address is: 3443 MARIPOSA ST, Bldg 3, Apt D

This is an example of four sub-addresses. The " 3443 MARIPOSA ST, Bldg 3" portion of the address is issued by the City Engineer. The "Apt D" portion of the address (known as the Sub-address) is not issue by the city engineer.

Note that the string "Apt" is not stored as part of the **ST_SUBADDR** value, but is stored in **ST_SUBTYPE**. The **ST_SUBTYPE** values will be stored in a valid value table to avoid inconsistencies.

Stored in database as:

ST_N	ST_SUF	ST_PRE	ST_NAM	ST_TY	ST_BLDGN
344			MARIPO	ST	3
344			MARIPO	ST	3
344			MARIPO	ST	3
344			MARIPO	ST	3

ADDITIONAL NOTES:

Some addresses and sub-addresses are stored in different fields although they look the same in an address string.

The address "123 MAPLE ST, Unit A" could represent an address issued by the City Engineer with an exterior entranceway, represented in the database as:

T_NU	ST_ FFIX	T_PREF	T_NAM	T_TY	ST LDGNUM	T_UNI	ST_S ADDR	S SUBTYPE
23			APLE	T				

The address "123 MAPLE ST, Unit A" could also represent an address with an interior entranceway. The "123 MAPLE ST" portion is issued by the City Engineer and the "Unit A" portion assigned by the owner. This would be represented in the database as:

T_NU	ST_ FFIX	T_PREF	T_NAM	T_TY	ST LDGNUM	T_UNI	ST_S ADDR	S SUBTYPE
23			APLE	T			A	U

There may be a building value without a unit value in the address.

There may be a unit value without a building value in the address.

A single building may contain both addresses and sub-addresses.

A correctly stored address record will never contain simultaneous values in the **ST_UNIT** and **ST_SUBADDR** fields in a correctly issued address.

■ Note: Applies to existing addresses only. New Addresses do not use this convention.

● Note: Required if address applies to a multi-building address location with assigned building numbers.

▲ Note: City Engineering does not currently maintain or issue apartment or condominium numbers. City Engineering does intend to provide for the storage and serving of apartment and condominium numbers.

b) Offset

Geocoding without an offset will locate addresses directly on the street centerline assuming a linear distribution of addresses along each street segment. If the locations are to be offset from the centerline an offset must be specified in the metadata provided with the database. The offset will be specified in state plane feet. Address locations will be offset perpendicular to the street centerline. All addresses will use the same offset.

2. Intersection Geocoding

Intersection geocoding can be used to spatially reference database information to specified intersections or intersection corners.

To permit geocoding of intersections the following requirements must be met.

a) Database Columns

The following columns must be present in the database:

ST1_FNAME

Description:	Full name for the first cross-street of the intersection.
Example:	W COLFAX AVE
Data type:	String
Width:	50
Includes:	The following sections of the street name should be specified, separated by single spaces. Refer to column definitions above (**Street Address Geocoding).** PREFIX (required if applicable) NAME (required) TYPE (required)

SUFFIX (required if applicable)
Req'd Value: Yes

ST2_FNAME
Description: Full name for the second cross-street of the intersection.
Example: N COLORADO BLVD
Data type: String
Width: 50
Includes: The following sections of the street name should be specified, separated by single spaces. Refer to column definitions above (**Street Address Geocoding).**
PREFIX (required if applicable)
NAME (required)
TYPE (required)
SUFFIX (required if applicable)
Req'd Value: Yes

The following additional columns must be present in the database table to map locations *to intersection corners*:

STX_OFFLEN
Description: Offset length applied from the center of the intersection in state plane feet (Refer to **Spatial Data Standards** for coordinate system information).
Example: 50
Data type: Numeric
Width: 4 (Maximum offset length 9999')
Valid Entries: Integer values 0-9999.
Req'd Value: Yes (may be 0 if no offset to be applied)

STX_OFFDIR
Description: Offset direction applied from the center of the intersection. Offset direction is applied relative to coordinate system north (Refer to **Spatial Data Standards** for coordinate system information).
Example: NW
Data type: String
Width: 2
Valid Entries: Cardinal directional designator in upper-case. Restricted to N, E, S, W, NE, NW, SE, SW. Do not include punctuation.

Req'd Value: Yes, if offset length is greater than 0. Otherwise the field should contain an empty string.

3. Coordinates

Coordinates can be used in a database to provide location. The coordinates are used to generate point geometry that is used to spatially reference the database.

The following standards are provided for storing coordinates in a database:

a) Database Columns

XCOORD

Description: Coordinate easting (or longitude for geographic coordinates)
Example: 3143301.48
Data type: Numeric
Width: Item width and number of decimal places dependent on coordinate system used and required precision
Notes: Geographic coordinates must be converted to decimal minutes to enable storage in numeric datatype.
Example:
104°30'00" W should be stored as -104.5

Note: The negative sign must be used for representing W longitude.

Note: The following table indicates the required significant figures in longitude decimal degrees for a variety of precisions in State Plane feet. One degree of latitude is approximately 69 miles in the Denver area.

Precision	**Decimal Degrees** (approx. significant figures required after decimal)
1 foot	6
10 foot	5
100 foot	4

YCOORD

Description: Coordinate northing (or latitute for geographic coordinates)
Example: 1694869.25
Data type: Numeric
Width: Item width and number of decimal places dependent on coordinate system used and required precision
Notes: Geographic coordinates must be converted to decimal minutes to enable storage in numeric datatype.
Example:
39°45'00" N should be stored as 39.75

Note: The following table indicates the required significant figures in latitude decimal degrees for a variety of precisions in State Plane feet for the Denver area. One degree of longitude is approximately 53¼ miles in the Denver area.

Precision	**Decimal Degrees** (approx. significant figures required after decimal)
1 foot	6
10 foot	5
100 foot	4

b) Metadata

The following spatial reference information (Refer to Metadata Standards – *This will be a link when the standards are completed and published*) must be included with the database to ensure the coordinate information is projected correctly to the City and County of Denver Standard Coordinate System (Refer to Spatial Data Standards – *This will be a link when the standards are completed and published*).

Spatial Reference

- Coordinate system
- Map projection
- Planar coordinate information
- Geodetic model

Data Quality Information

- Source information
- Process information (method of capture)

- Accuracy

Note: Storing and providing data in the City and County of Denver Standard Coordinate System is highly recommended where possible.

IV. Database Formats

Choosing a database system/format that will support business needs is an important step in developing a database that is to be spatially referenced. The database system chosen must be appropriate to the required usage, update frequency, availability and integrity of the data. A list of standard database formats is provided along with appropriate uses for each database. The database systems are grouped into relational and non-relation databases in order of recommendation (most recommended at top of list).

Please note that there are many combinations of databases and applications that may be appropriate to meet business needs. It is highly recommended that initial plans are reviewed with Corporate GIS before investing significant planning for a particular database. This includes databases that seem very simple to implement (example: Access databases or Excel spreadsheets), many of which have longer-term maintenance and update implications that are not initially perceived.

A. Relational Database Management Systems (RDBMS)

Relational databases are suited for information in which relationships can be defined that enable improved storage and maintenance over non-relational databases. Relational databases provide capabilities to enforce relationships, thereby encouraging information integrity, improved querying and reporting, and elimination of duplicate information. In general, a relational database should be chosen for databases that are frequently updated and/or frequently exported or linked to spatial information.

1. Oracle

Summary:
- Server database management system

Recommendation:
- Recommended DBMS. This is the DBMS used to support the Corporate GIS Spatial Database Engine (SDE). Using this format provides the greatest level of interoperability between spatial data and the database.

Suitability:
- Suitable for managing large complex databases.

- Suitable for frequent (or real-time) update integration with GIS
- Will enable integration with all Corporate GIS applications (present and future) including desktop, wireless, intranet and internet applications.
- Suitable for implied spatial referencing in which the database contents will be used to generate spatial features

Required to:
- Link databases directly (real-time) to features in SDE for mapping in Corporate GIS applications (intranet, internet).

2. Microsoft SQL Server

Summary:
- Server database management system

Recommendation:
- Recommended DBMS where experience or support not available for maintaining Oracle databases.
- Highly recommended over Access to encourage data integrity and access control.

Suitability:
- Suitable for managing large complex databases.
- Suitable for infrequent (monthly, quarterly, annually) update integration with GIS. *Note that database contents must be replicated to Oracle to be able to link features directly without additional application programming. This means that real-time data updates are not possible if the information is stored in SQL Server.*
- Suitable for implied spatial referencing in which the database contents will be used to generate spatial features

Required to:
- Link databases to SDE by replicating contents from SQL Server to SDE for mapping in Corporate GIS applications (intranet, internet).

3. Microsoft Access (2000 or greater)

Summary:

- File based database management system

Recommendation:
- Only recommended for database prototyping or internal management of non-production databases. Will not be used to support production GIS applications.

Suitability:
- Suitable for managing small, simple databases
- Suitable for one-time or infrequent (bi-annually, annually) update integration with GIS. Note that database contents must be replicated to Oracle to be able to link features directly without additional application programming.
- Suitable for implied spatial referencing in which the database contents will be used to generate spatial features
- Not suitable for key-based spatial referencing unless keys are used to generate spatial features

Required for:
- Personal geodatabases (refer to Spatial Data Standards). *Personal geodatabases are a new model for storing spatial data on workstations using ArcGIS. Standards for personal geodatabases are provided in Spatial Data Standards.*

B. Non-Relational Databases

Non-relational databases are best suited for information that can be represented in tabular format (single table) without significant duplication of data. Non-relational databases are best suited for small databases that are managed by a small number of individuals. Maintenance guidelines must be established and strictly followed to ensure database integrity.

1. dBase (.dbf)

Summary:
- Tabular database stored in .dbf file. Basic datatypes and columns definitions supported.

Recommendation:
- To be used only where suitable. See Suitability
- Not recommend as a format for data maintenance and updates.

Suitability:

- Most suitable for use as an export format for providing infrequent or one-time updates of data that will be spatially referenced and loaded into the Corporate Spatial Database.
- The dBase format is used for storing attributes in ESRI Shapefiles and should be provided with Shapefiles submitted to Corporate GIS. Shapefile standards are detailed in Spatial Data Standards.

Required to:
- Provide attributes with Shapefiles (see Spatial Data Standards)

2. ASCII (tab or comma delimited)

Summary:
- Tabular database stored in ASCII text file. Intended datatypes and column definitions must be provided in metadata to ensure correct interpretation.

Recommendation:
- To be used only where suitable. See Suitability
- Not recommend as a format for data maintenance and updates.

Suitability:
- Most suitable for use as an export format for providing infrequent or one-time updates of data that will be spatially referenced and loaded into the Corporate Spatial Database.
- ASCII data is typically used as a lowest common denominator format to enable transfer of data between different systems. It is important that the ASCII data is formatted appropriately and that complete metadata is provided to ensure that the data is loaded correctly. It is recommended that the specific formatting chosen (comma separated values, tab-delimited etc.) be reviewed with Corporate GIS to ensure the data will be represented and transferred correctly.

Required for:
- May be required to transfer data from legacy systems

3. Microsoft Excel (.xls)

Summary:
- Tabular data stored in .xls file

Recommendation:
- To be used only where suitable. See Suitability

- Not recommend as a format for data maintenance and update.

Suitability:
- Not recommended as a storage system for managing data that will be spatially referenced. If Excel is to be used simply because Microsoft Office is available, it is preferred that Access be used.

Required for:
- None

V. Database Submittal Requirements

A. Maintenance Plan

A database maintenance plan must be submitted if the database will be maintained external to the Corporate GIS and will be updated (spatially referenced) on a regular basis.

The maintenance plan ensures that roles and responsibilities are clearly defined and that regular scheduled updates can be coordinated and performed consistently.

The maintenance plan should include the following:

1) Description of the database
2) Intended use
3) Database location (server name, connection info, path etc.)
4) Support personnel
5) Update procedures
6) QA/QC procedures
7) Required GIS update schedule
8) Backup plan

B. Database Design

1. Relational Design Guidelines

The following guidelines are provided for the design of relational databases. For more information please contact Corporate GIS.

a) Normalization

It is recommended that databases be normalized to 3rd normal form, according to the following:

a) There are no duplicated rows in a table.

b) There are no multi-valued attributes in an item.

Example of non-normalized form:

USERS
Bob,Lucy,Mary

c) There are no repeating groups of attributes.

Example of non-normalized form:

USER1	USER2	USER3
Bob	Lucy	Mary

d) All attributes should be dependent upon the key

Example of non-normalized form:

CLASS_ID	CLASS_NAME	STUDENT_NAME
123	GIS 101	Bob

In this example, the student name (non-key attribute) is not directly dependent on the key (CLASS_ID). Only class attributes should be dependent on the CLASS_ID key and the student information should be moved to a STUDENTS table.

2. Documentation

A design document must be submitted for all databases. The design document will include a data dictionary and an entity-relationship diagram (ERD) when a relational database is used.

a) Data Dictionary

The Data Dictionary should describe all tables and data elements in the database as follows:

a) Tables
   1. Name (Real life name)
   2. Description (Short description)
   3. Key (Key column(s))
   4. Restrictions (Security restriction if applicable)
   5. Notes (Additional notes)

b) Columns

1. Name (Real life name)
2. Description (Short description)
3. Restrictions (Security restrictions if applicable)
4. Format (Data type, size)
5. Null allowed (Null values allowed)
6. Allowed Values (Values allowed)
7. Default (Default value)
8. Source (Source of data entry)
9. Key (Is primary or foreign key?)
10. Indexed (Indexed column?)
11. Notes (Additional notes)

A Data Dictionary sample is provided in Appendix A.

b) ERD (Entity-Relationship Diagram)

An entity-relationship diagram shall be submitted that meets the following requirements:

c) Completed diagram shall be submitted digitally in ERWin (.erx) or Visio (.vsd) format. ERDs created in Microsoft Visio 2000 or greater are preferred.
Note: VisioModeler 3.1, available free from Microsoft can be used to create ER Diagrams that can be exported to .erx format. VisioModeler is available at *http://www.microsoft.com/downloads/release.asp?ReleaseID=27489*

d) A hardcopy printout should be also provided and included with the Database Documentation

e) The ERD shall:
   i. Identify all table and view names
   ii. Identify all column names and definitions (datatypes, widths)
   iii. Identify all primary and foreign keys
   iv. Identify all relationships (including appropriate verbs) using Crow's Foot notation. Ordinality and cardinality must be defined in the design.

An ERD sample is provided in Appendix B.

VI. Entity Naming

A. Table and View Names

1. Constraints

- Name must be 30 characters or less
- Must start with a letter and may contain only letters, digits and the underscore (_) character. Spaces cannot be used.
- Name should be in uppercase with underscores used as word separators.

2. Plurality

- When the table represents information about a topic or business area the name should be singular (examples: CONFIGURATION, FEEDBACK, INFRASTRUCTURE)
- When the table represents information about entities contained in the table the name should be plural (examples: ROADS, LOCATIONS, USERS).
- When the table represents information about a single entity (and is never expected to contain more than one) it may be appropriate to use a singular name (example: AREA_OF_INTEREST vs. AREAS_OF_INTEREST when only one area of interest)

3. Guidelines

- Names should be spelled out as far as possible for the database being used.
- Commonly recognized abbreviations should be used where appropriate (Refer to Naming Standards)
- Use consistent abbreviations across all tables in a database

B. Column Names

1. Constraints

- Name must be 30 characters or less
- Must start with a letter and may contain only letters, digits and the underscore (_) character. Spaces cannot be used.
- Name should be in uppercase with underscores used as word separators.

2. Guidelines

- Names should be spelled out as far as possible for clarity.
- Commonly recognized abbreviations should be used where appropriate (Refer to Naming Standards)
- Use consistent abbreviations across all items in all database tables
- Primary and foreign keys should have the same name.
- Primary key names should be consistent with the table name. (Example: The primary key in the ROADS table should be named ROADS_ID).

VII. Appendix A – Data Dictionary Sample

Maps Database Data Dictionary

Monday, August 18th, 2003

Table: MAPS

Name:	Maps
Description:	Represents all available maps
Key:	MAP_ID
Restriction:	None
Notes:	All maps available should be listed in this table

Column: MAP_ID

Name:	Map Identifier
Description:	Auto-numbered identifier for each map
Restrictions:	None
Format:	Long Integer
Null Allowed:	N
Allowed Values:	1 to 2,147,483,647
Default Value:	Auto-increment from previous value
Source:	Auto-numbered
Key:	Primary
Indexed:	Yes
Notes:	This number entered automatically when new records added

Column: NAME

Name:	Map Identifier
Description:	Name (title) of the map
Restrictions:	None
Format:	String (100)
Null Allowed:	N
Allowed Values:	Any string value
Default Value:	
Source:	User input
Key:	No
Indexed:	No
Notes:	Name should match title as shown on map

(etc…)

Note: This is a fictitious database for sample purposes only

VIII. Appendix B – Entity-Relationship Diagram (ERD) Sample