# Mining Mountains of Data: Organizing All Atom Molecular Dynamics Protein Simulation Data into SQL and OLAP Cubes

Andrew M. Simms
Daggett Lab

# Overview

- Background
  - Daggett Lab
  - On-line Analysis Processing (OLAP)
- OLAP Details
- An OLAP Cube Design for Simulations
- Implementation Results
- Conclusions

Daggett Lab

# BACKGROUND

# Daggett Lab

- Two areas of focus
  - Disease related proteins
  - Dynameomics
- Primarily computational
- Both focus areas study protein motion

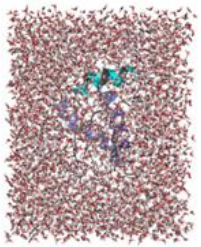
www.dynameomics.org


Van der Kamp MW et al. Dynameomics: A comprehensive database of protein dynamics. **Structure**, 18: 423-435, 2010. [DOI]

# Molecular Dynamics (MD)

- Atomic resolution structure and dynamics


Build


Solvate

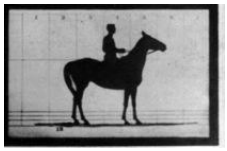

10ns

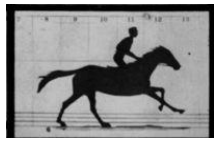

20ns

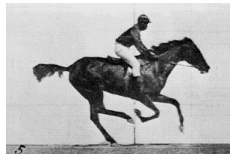

30ns


40ns


50ns

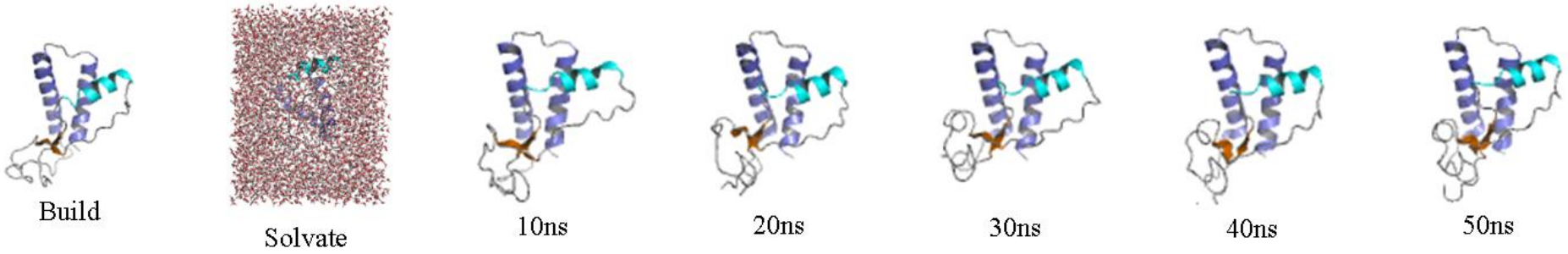






| sim_id | struct_id | struct_inst | atom_number | step | x_coord | y_coord | z_coord | bin |
|---|---|---|---|---|---|---|---|---|
| 678 | 122 | 1 | 1 | 0 | -5.846 | 8.722 | 11.445 | 408 |
| 678 | 122 | 1 | 2 | 0 | -5.989 | 8.026 | 12.191 | 480 |
| 678 | 122 | 1 | 3 | 0 | -4.842 | 8.797 | 11.24 | 408 |
| 678 | 122 | 1 | 4 | 0 | -6.157 | 9.627 | 11.775 | 480 |
| 678 | 122 | 1 | 5 | 0 | -6.634 | 8.372 | 10.247 | 408 |

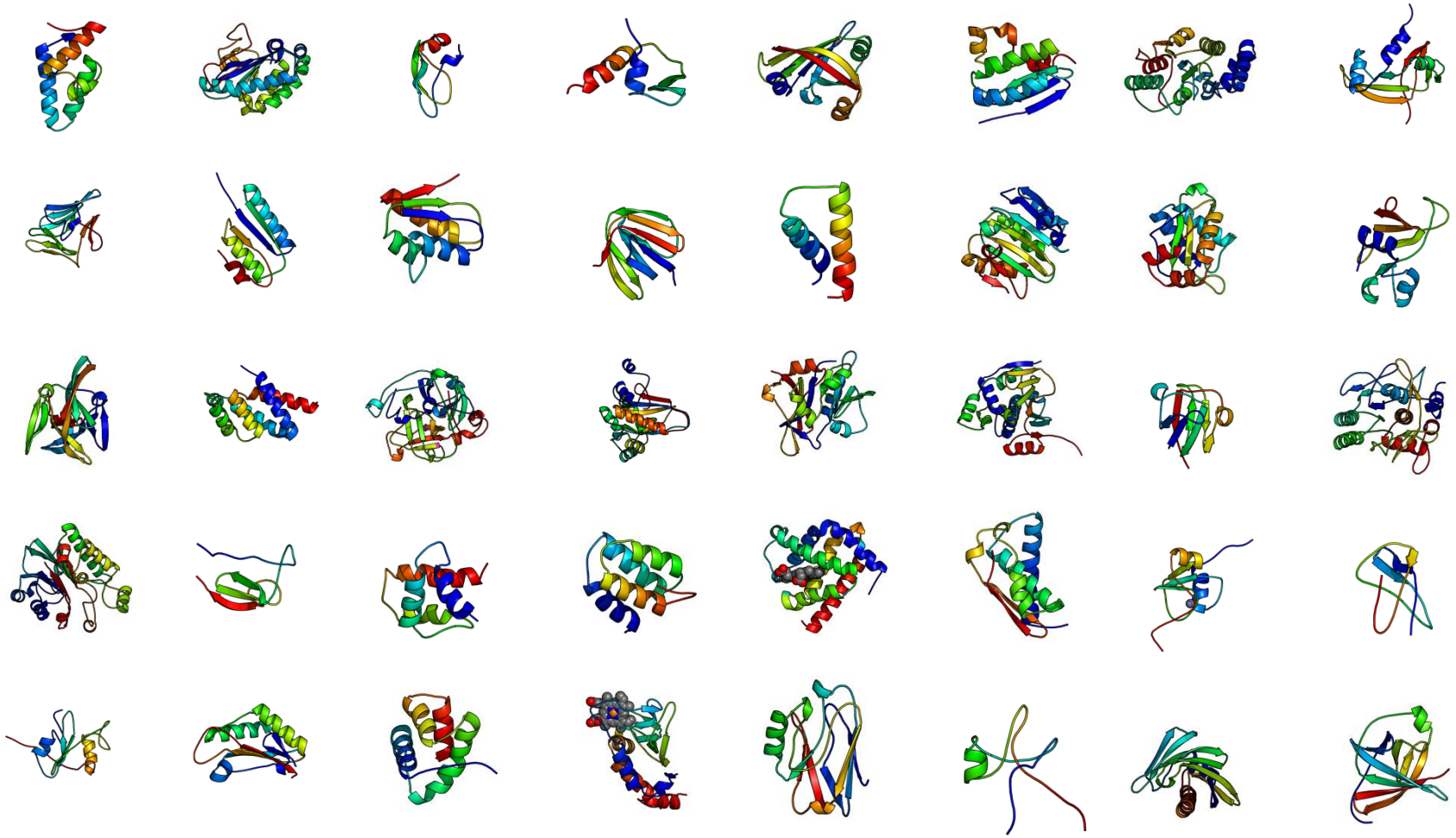
# One Simulation




- A “typical” simulation contains

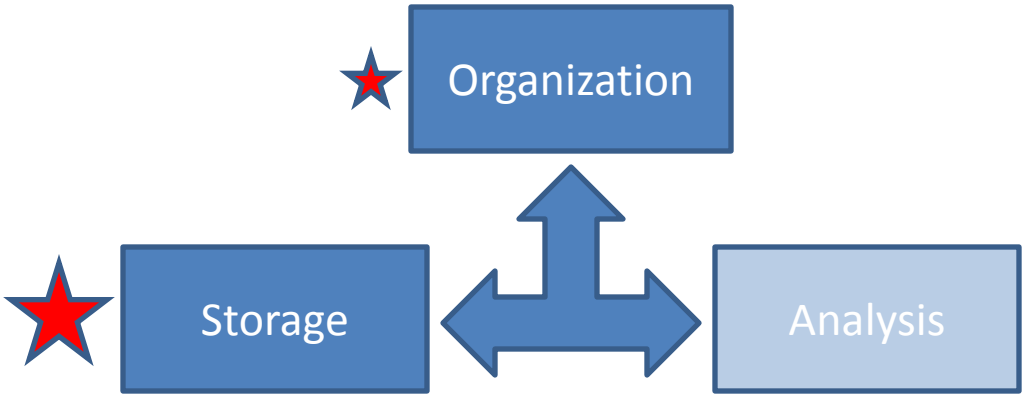
| Protein Coordinates | Structures | Coordinate Table | Analysis Tables |
|---|---|---|---|
| $29.3 \times 10^6$ | $31.0 \times 10^3$ | 4.4GB | 0.6GB |

# 2,070 Targets Simulated



# Informatics Challenge (in 2007)

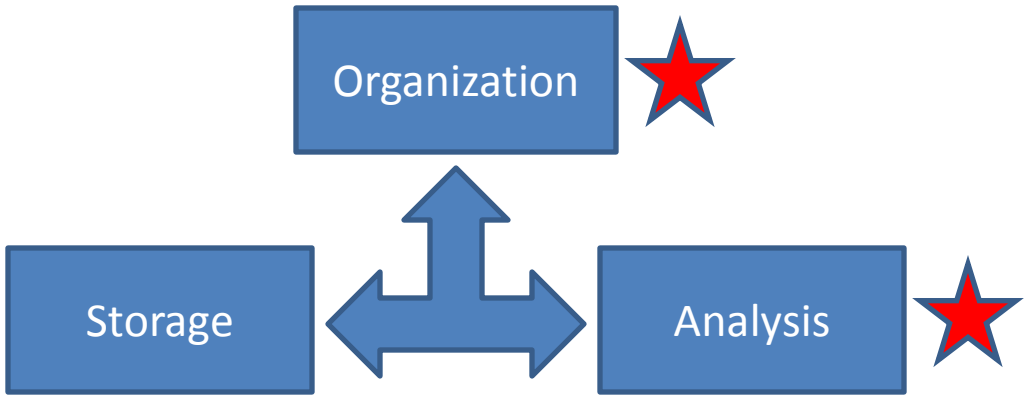
- Storage and basic organization




| Simulations | Targets | Time | Structures | SQL |
|---|---|---|---|---|
| 2,300+ | 300+ | 35 μs | 50,600,000 | **~24 TB** |

# Informatics Challenge Now

- The lab has run over 10,915 simulations, each containing millions to billions of protein atom coordinates and even more analyses




| Simulations | Proteins | Time | Structures | Space |
|---|---|---|---|---|
| 7,344+ | 1248+ | 186 μs | 251 x $10^6$ | 71+ TB |

BACKGROUND

# ONLINE ANALYSIS PROCESSING (OLAP)

# Online Analytical Processing (OLAP)

- Term coined by Ted C. F. Codd, the inventor of the relational data model
- Described as a set of principals that posit the type of database needed for transactional tasks is fundamentally different than the type of database needed for analysis

Codd EF, Codd SB, Salley CT. Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate 1993.

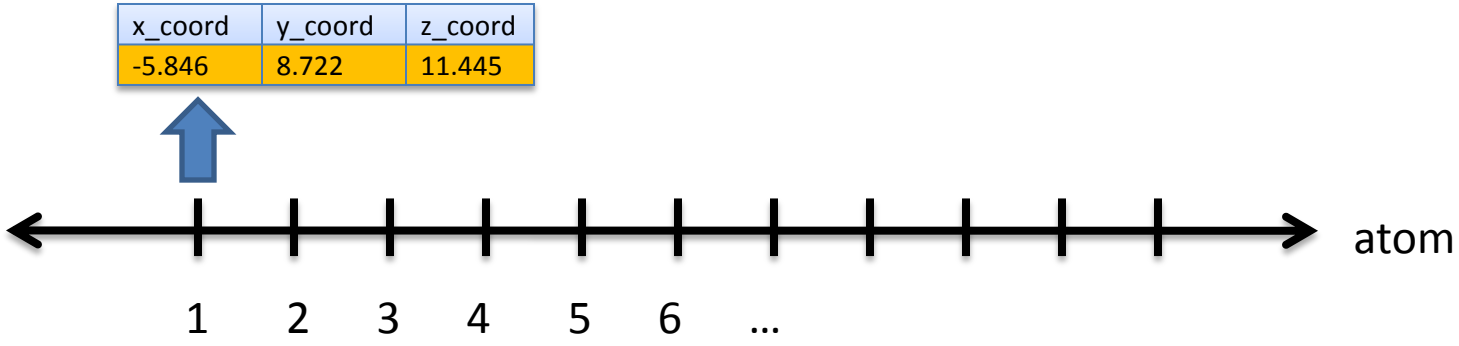
# OLAP Concepts

- Data are organized around **FACTS** and **DIMENSIONS**
- **FACTS** are continuous measurements on a item of interest
- **DIMENSIONS** are discrete quantities that classify measurements into useful groupings

| sim_id | struct_id | struct_inst | atom_number | step | x_coord | y_coord | z_coord | bin |
|---|---|---|---|---|---|---|---|---|
| 678 | 122 | 1 | 1 | 0 | -5.846 | 8.722 | 11.445 | 408 |
| 678 | 122 | 1 | 2 | 0 | -5.989 | 8.026 | 12.191 | 480 |
| 678 | 122 | 1 | 3 | 0 | -4.842 | 8.797 | 11.24 | 408 |
| 678 | 122 | 1 | 4 | 0 | -6.157 | 9.627 | 11.775 | 480 |
| 678 | 122 | 1 | 5 | 0 | -6.634 | 8.372 | 10.247 | 408 |

# Dimensions

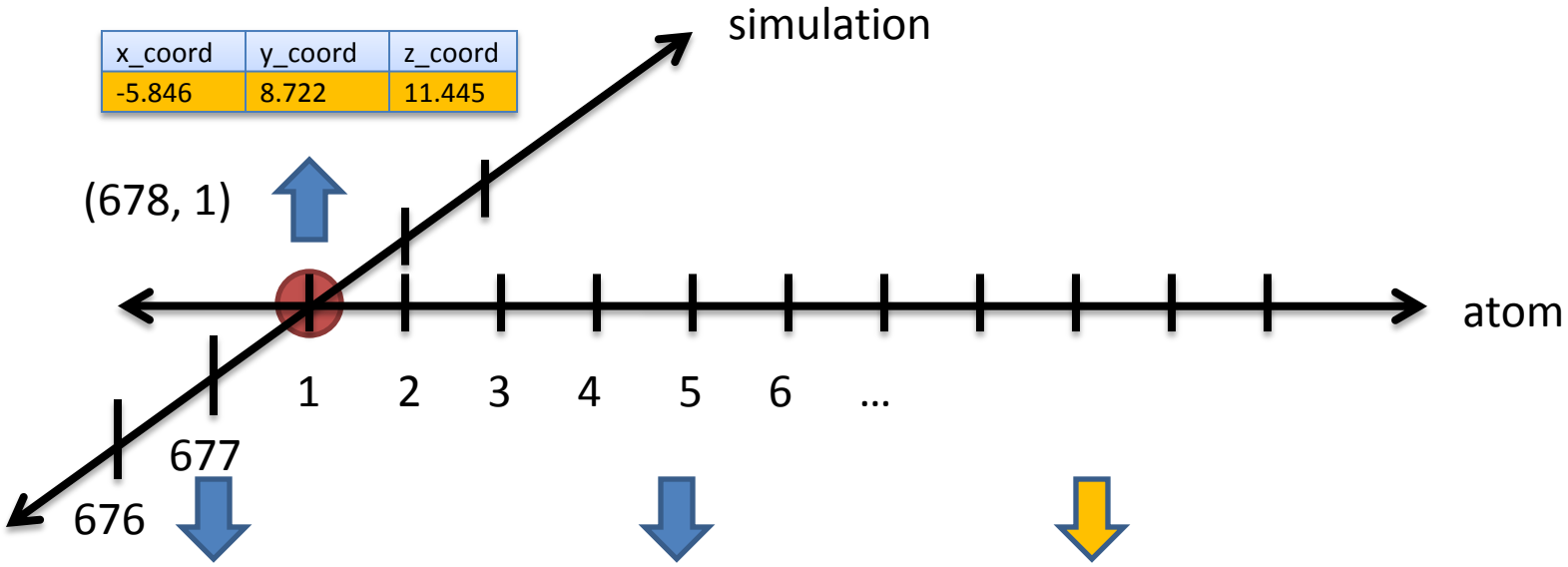
- An individual dimension is similar to a number line, but you are not limited to integers




| sim_id | struct_id | struct_inst | atom_number | step | x_coord | y_coord | z_coord | bin |
|---|---|---|---|---|---|---|---|---|
| 678 | 122 | 1 | 1 | 0 | -5.846 | 8.722 | 11.445 | 408 |
| 678 | 122 | 1 | 2 | 0 | -5.989 | 8.026 | 12.191 | 480 |
| 678 | 122 | 1 | 3 | 0 | -4.842 | 8.797 | 11.24 | 408 |
| 678 | 122 | 1 | 4 | 0 | -6.157 | 9.627 | 11.775 | 480 |
| 678 | 122 | 1 | 5 | 0 | -6.634 | 8.372 | 10.247 | 408 |

# Dimensions, Continued

- A set of dimensions provide coordinates to facts

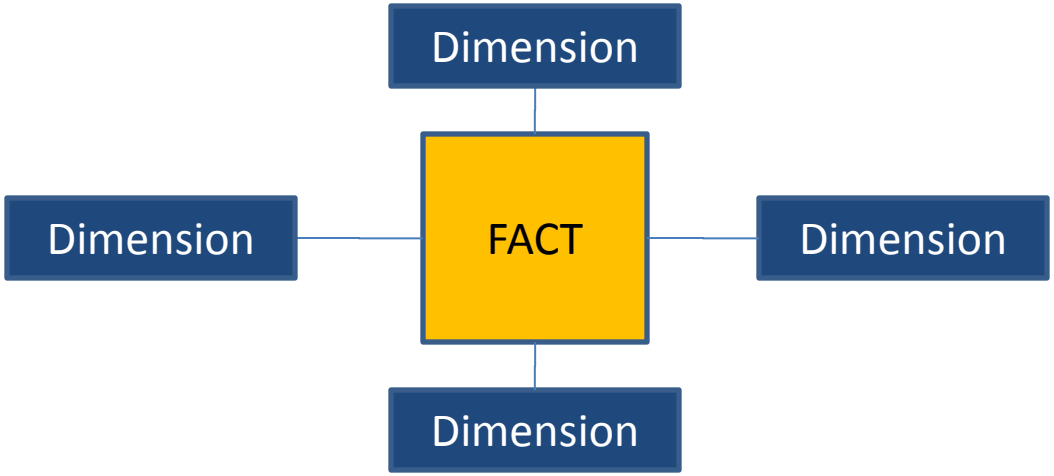



| x_coord | y_coord | z_coord |
|---|---|---|
| -5.846 | 8.722 | 11.445 |

| sim_id | struct_id | struct_inst | atom_number | step | x_coord | y_coord | z_coord | bin |
|---|---|---|---|---|---|---|---|---|
| 678 | 122 | 1 | 1 | 0 | -5.846 | 8.722 | 11.445 | 408 |
| 678 | 122 | 1 | 2 | 0 | -5.989 | 8.026 | 12.191 | 480 |
| 678 | 122 | 1 | 3 | 0 | -4.842 | 8.797 | 11.24 | 408 |
| 678 | 122 | 1 | 4 | 0 | -6.157 | 9.627 | 11.775 | 480 |
| 678 | 122 | 1 | 5 | 0 | -6.634 | 8.372 | 10.247 | 408 |

# OLAP Cubes

- A collection of facts and related dimensions form a (hyper) cube




- The cube concept can be implemented using relational tables in a star schema or using a multi-dimensional database…

# Multidimensional OLAP

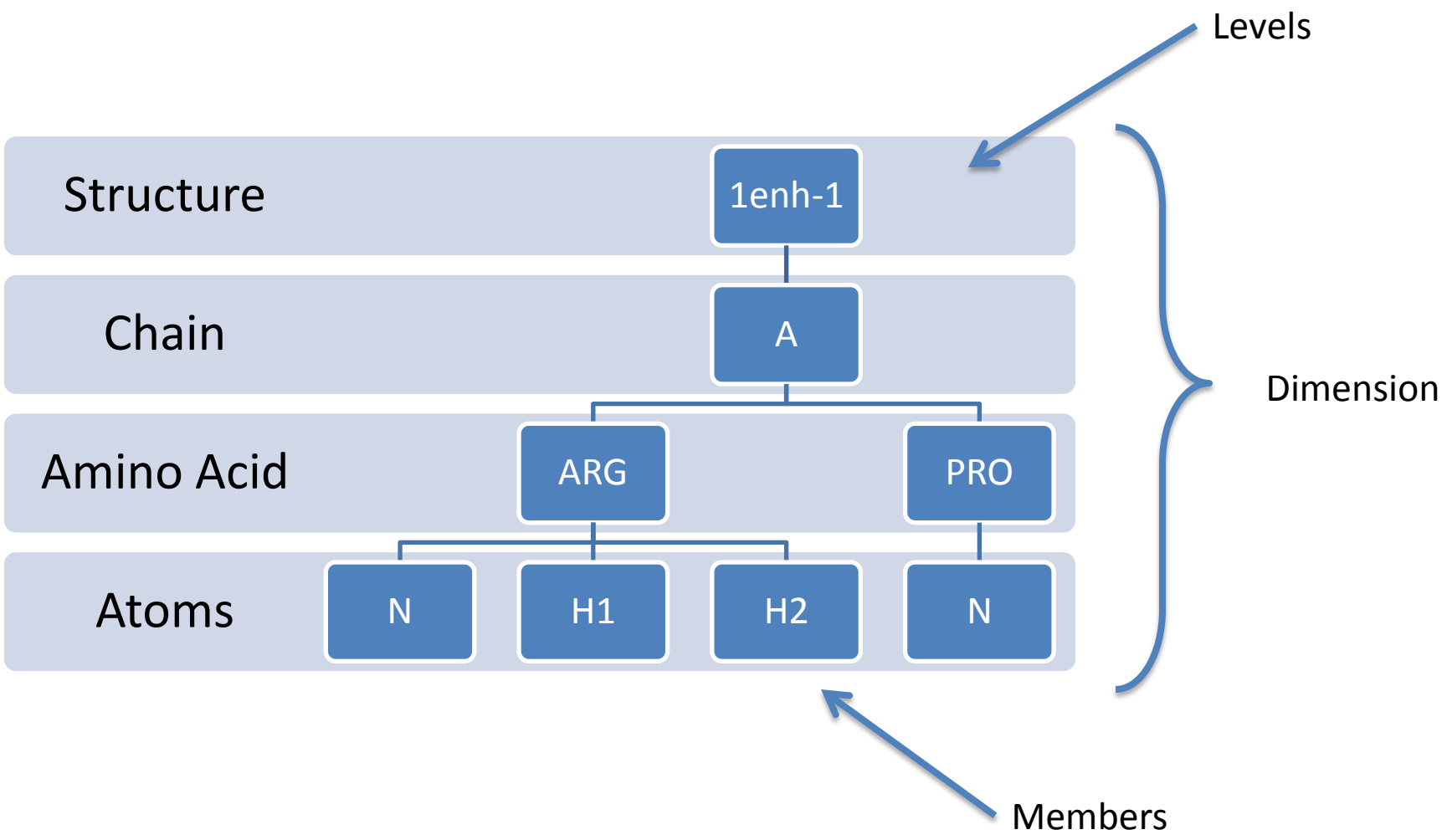
- MOLAP is an implementation of a OLAP database optimized for multidimensional storage
- SQL Server Analysis Services (SSAS) is a set of tools including a MOLAP storage engine and the Multi-Dimensional Expressions (MDX) language

# OLAP IN DETAIL

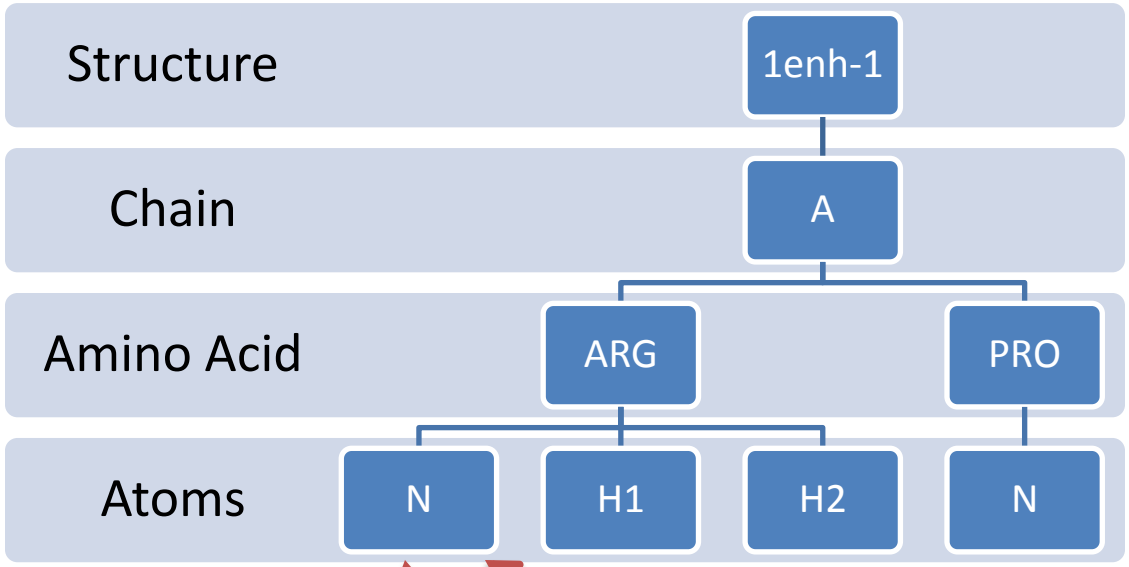
# Dimensions

- Recall dimensions uniquely identify facts
- Dimensions are composed of discrete values called **members**
- Fact data can be “addressed” by specifying a member from each associated dimension
- Members can be organized in a hierarchy

# Hierarchies



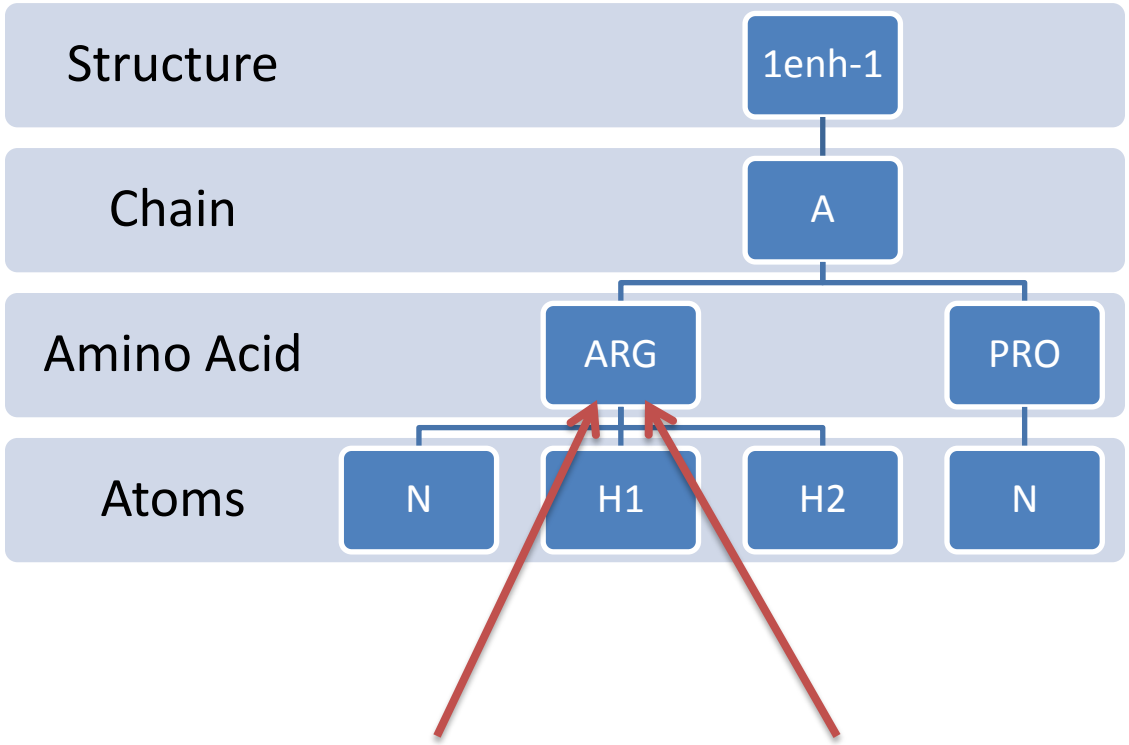
# Facts are Associated with Members




- Example: coordinates and atoms in a structure

| sim_id | struct_id | struct_inst | atom_number | step | x_coord | y_coord | z_coord | bin |
|---|---|---|---|---|---|---|---|---|
| 678 | 122 | 1 | 1 | 0 | -5.846 | 8.722 | 11.445 | 408 |
| 678 | 122 | 1 | 2 | 0 | -5.989 | 8.026 | 12.191 | 480 |
| 678 | 122 | 1 | 3 | 0 | -4.842 | 8.797 | 11.24 | 408 |
| 678 | 122 | 1 | 4 | 0 | -6.157 | 9.627 | 11.775 | 480 |
| 678 | 122 | 1 | 5 | 0 | -6.634 | 8.372 | 10.247 | 408 |

# Facts can be associated members at any level




- Example: Dihedral angles are computed on amino acids

| sim_id | struct_id | struct_inst | residue_id | step | dh_id | dh_angle |
|---|---|---|---|---|---|---|
| 678 | 122 | 1 | 2 | 0 | 1 | 5.412 |
| 678 | 122 | 1 | 2 | 0 | 2 | -1.562 |
| 678 | 122 | 1 | 2 | 0 | 5 | -2.908 |
| 678 | 122 | 1 | 2 | 0 | 8 | 6.536 |

# Tuples and Sets

- A **tuple** is the collection of dimension members that define a fact
- Similar to a multidimensional array in C#
  - Float[,,,,,] myarray = new Int32[10000, 2400, 50, 900, 300000,255];
  - myarray[678,122,1,2,0,1] = 5.412
- Unlike a C# array, OLAP dimensions are self describing and can listed in any order
- A **set** is a collection of **tuples**

| sim_id | struct_id | struct_inst | residue_id | step | dh_id | dh_angle |
|---|---|---|---|---|---|---|
| 678 | 122 | 1 | 2 | 0 | 1 | 5.412 |
| 678 | 122 | 1 | 2 | 0 | 2 | -1.562 |
| 678 | 122 | 1 | 2 | 0 | 5 | -2.908 |
| 678 | 122 | 1 | 2 | 0 | 8 | 6.536 |

# OLAP and Aggregation

- Individual facts are specified by a “tuple”
- Leaving out a dimension means “*” or all, resulting in a set
- Choosing a member above the base is short hand for a set of all descendants
- OLAP will apply the defined aggregation, typically SUM

| sim_id | struct_id | struct_inst | residue_id | step | dh_id | dh_angle |
|---|---|---|---|---|---|---|
| 678 | 122 | 1 | 2 | 0 | 1 | 5.412 |
| 678 | 122 | 1 | 2 | 0 | 2 | -1.562 |
| 678 | 122 | 1 | 2 | 0 | 5 | -2.908 |
| 678 | 122 | 1 | 2 | 0 | 8 | 6.536 |

# OLAP is not for Managing Data

- OLAP cubes **do not**
  - care about integrity constraints
  - support easy or fast updates to data
  - worry about missing or sparse data
- One way to think of OLAP – a materialized and optimized view of data stored somewhere other than the store of record (which is typically SQL)

# Microsoft SQL Server: OLAP and Relational

**Analysis Services (MOLAP)**

- Cube
- Proprietary Store*
- Language is MDX
- Queries are top-down
- Results are multi-dimensional cubes
- Data are ORDERED

**SQL Server (Relational)**

- Database
- Relational Store
- Language is SQL
- Queries are bottom-up
- Results are two-dimensional tables
- Data are UNORDERED

* No longer undocumented: I Gorbach, A. Berger, E. Melomed, Microsoft SQL Server 2008 Analysis Services UNLEASHED, 2009 Pearson Education, Inc.

# SQL Server Analysis Services

- Discrete dimensional values mean indexes can be implemented as bit vectors—fast but difficult to update and create
- Data are inherently ordered, making it easy to do things like compute medians
- Cubes effectively must be compiled from other sources

# Multidimensional Expressions (MDX)

- The query language for Analysis Services is MDX
- An MDX query defines a sub-cube, possibly multi-dimensional, derived by slicing and dicing (their words) data from the source cube

# A Quick Look at MDX

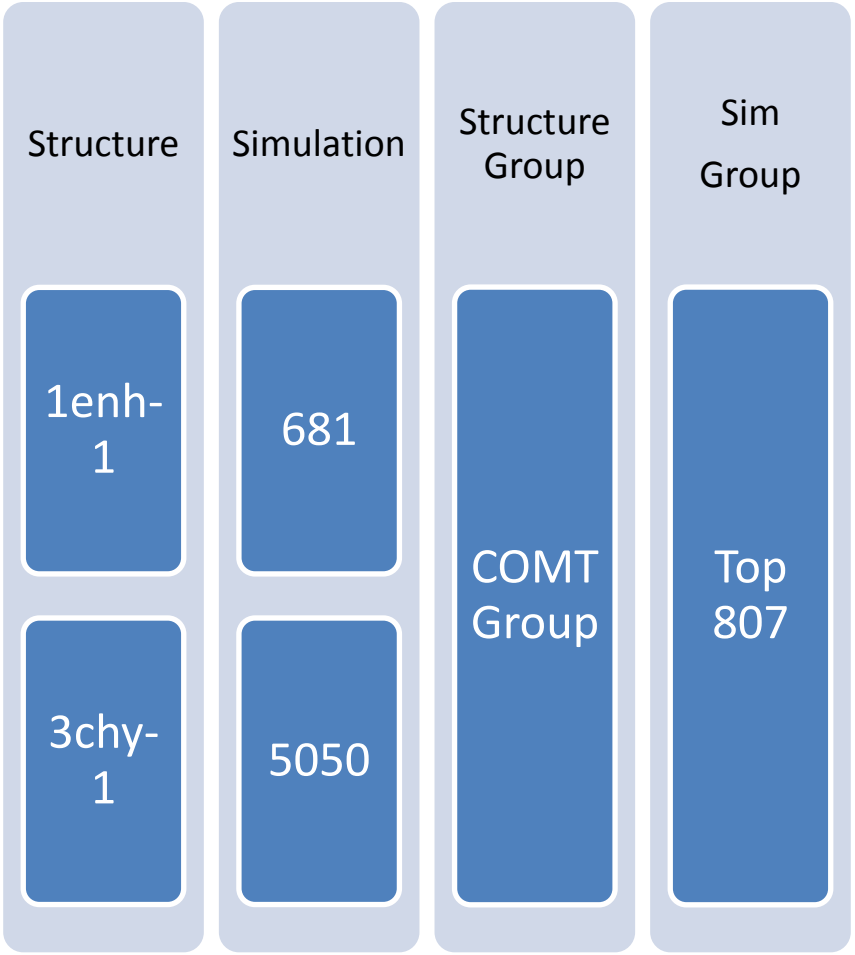
```
WITH
MEMBER [Measures].[atm_type] as '[Structure].[Atom Type].membervalue'
MEMBER [Measures].[res_type] as '[Structure].residue.membervalue'
MEMBER [Measures].[res_num] as '[Structure].[residue number].membervalue'
MEMBER [Measures].[atm_num] as '[Structure].[Structure Hierarchy].Properties("Atom Number")'

SELECT {      [Measures].[atm_num]
                          , [atm_type]
                          , [res_type]
                          , [res_num]
                          , [x Coord]
                          , [y Coord]
                          , [z Coord] } on AXIS(0)
              , { [Structure].[Structure Hierarchy].[Atom].&[122]&[1] :
                  [Structure].[Structure Hierarchy].[Atom].&[122]&[5] } on AXIS(1)
FROM [UnifiedDSV]
WHERE ( [Simulation].[Simulation Hierarchy].[Step].&[678]&[1]&[0] )
```

| | atm_num | atm_type | res_type | res_num | x Coord | y Coord | z Coord |
|---|---|---|---|---|---|---|---|
| N | 1 | N | ARG | 1 | -5.846 | 8.722 | 11.445 |
| H1 | 2 | H | ARG | 1 | -5.989 | 8.026 | 12.191 |
| H2 | 3 | H | ARG | 1 | -4.842 | 8.797 | 11.24 |
| H3 | 4 | H | ARG | 1 | -6.157 | 9.627 | 11.775 |
| CA | 5 | C | ARG | 1 | -6.634 | 8.372 | 10.247 |

# A CUBE DESIGN FOR SIMULATIONS

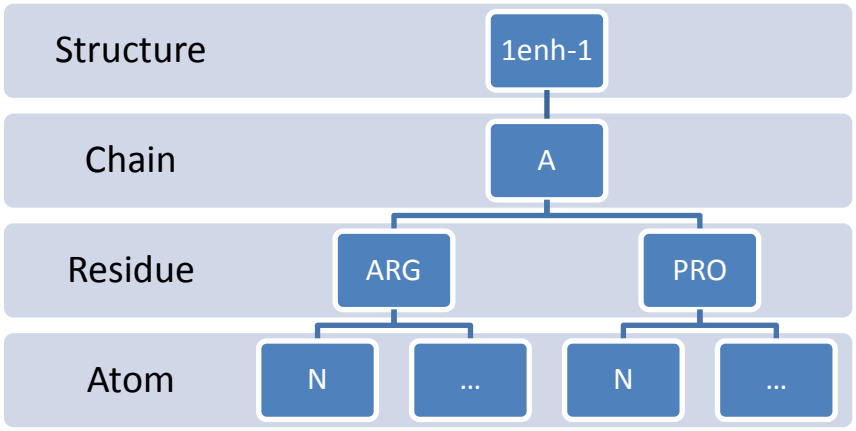
# Design

- Dynameomics has 4 OLAP dimensions
  - Structure
  - Simulation
  - Simulation Group
  - Structure Group

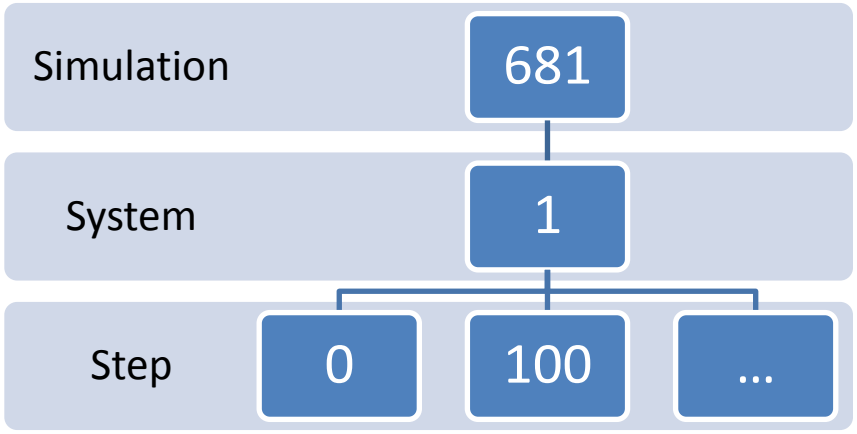


# Primary Dimensions

## Structure Dimension



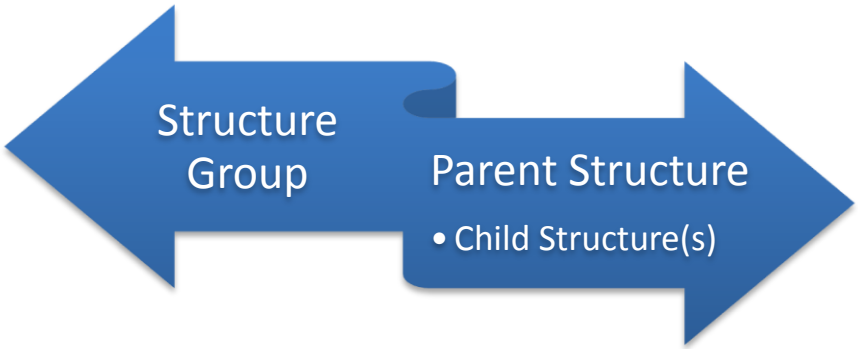

## Simulation Dimension

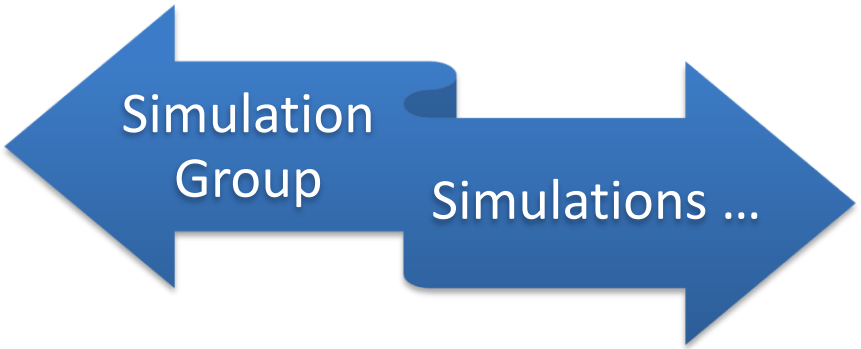


# Secondary Dimensions

**Structure Group is a Many-to-Many relationship between Structures**

**Simulation Group is a Many-to-Many relationship with Simulations**



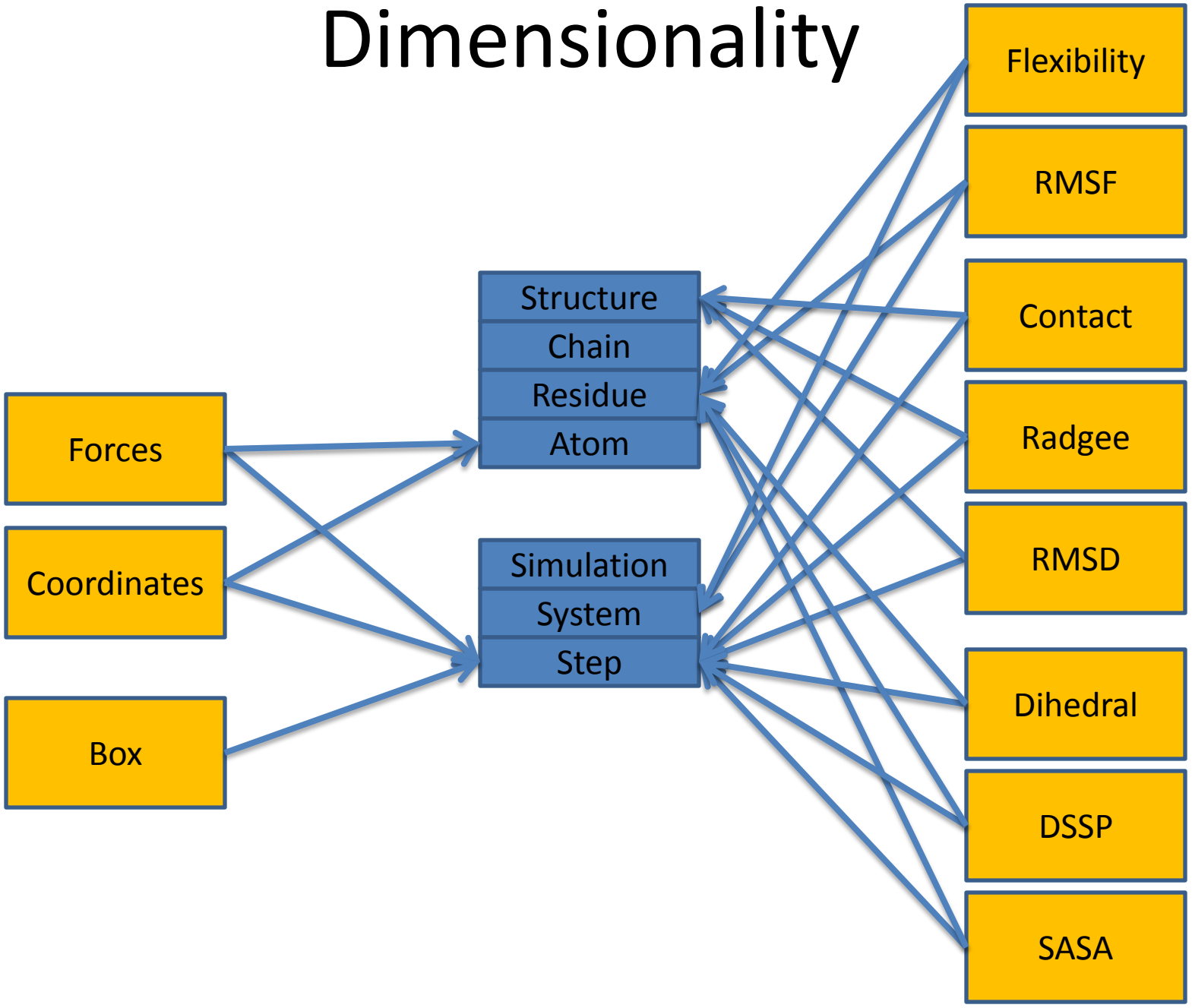
# Dimensions and SQL

- Dimensions are closely tied to SQL tables in the main warehouse
  - **Simulation** is keyed at the lowest level on sim_id, struct_inst, struct_id and step
  - **Structure** is keyed at the lowest level on struct_id and atom_number
  - **Structure Groups** and **Simulation Groups** are related though intermediate tables to Structure and Simulation, respectively

# Facts (Measures)

- **Atom Coordinates**
- **Box**
- Forces
- **Dihedral Angles**
- DSSP
- Flexibility
- Solvent Accessible Surface Area (SASA)
- Cα RMSD
- Congenial
- Radius of Gyration (Radgee)
- Contacts

# Dimensionality



# IMPLEMENTATION

# Starting Point

- A cube based on the top 6 Dynameomics targets:

| Simulations | Structures | Time | SQL Space |
|---|---|---|---|
| 63 | $1.5 \times 10^6$ | 1.15 μs | 214GB |

- Contains coordinates, box size, and dihedral angles only

# Initial Observations

- Initial build/processing time ~2 hours
- Cubes are significantly smaller than their SQL counter parts:

| Simulations | Structures | Time | SQL Space | OLAP Space |
|---|---|---|---|---|
| 63 | $1.5 \times 10^6$ | 1.15 μs | 214GB | **41GB** |

# Test Query: Dihedral Angles

- Dihedral angles are used to study side-chain conformations
- One visualization technique is to make histograms, effectively binning observed angles into 1 degree buckets
- 855,484,304 rows of Dihedral data in my test set
- In SQL…

# SQL Dihedral Query

```
SELECT      byres.residue
            , dh.angle_name
            , byres.[bin]
            , SUM(byres.[count]) AS [count]
FROM (
            SELECT      i.residue
                        , d.dh_id
                        , CAST (ROUND(d.dh_angle,0) AS INT) AS [bin]
                        , COUNT(*) AS [count]
            FROM        ( SELECT DISTINCT struct_id, residue_id, residue
                          FROM [Directory].dbo.Master_ID  ) AS i
                        JOIN [dynameomics-9].dbo.andrew_TOP6_Dihed AS d WITH (NOLOCK)
                        ON ( i.struct_id = d.struct_id
                                    AND i.residue_id = d.residue_id )
                        GROUP BY i.residue
                                    , d.dh_id
                                    , CAST (ROUND(d.dh_angle,0) AS INT)
                        UNION ALL
                        -- FOUR MORE SELECTS HERE
            ) AS byres
            JOIN dbo.Dihedral_Angle AS dh
            ON ( byres.dh_id = dh.dh_id )
GROUP BY    byres.residue , dh.angle_name, byres.[bin]
ORDER BY    byres.residue, dh.angle_name, byres.[bin]
```

# SQL Results

- First version was too slow (I stopped it after 3 hours)
- Second version, 65 lines, took 32 minutes, 35,364 rows
- This query could be more thoroughly analyzed and perhaps made faster

| residue | angle_name | bin | count |
|---|---|---|---|
| ALA | chi1 | -180 | 53596 |
| ALA | chi1 | -179 | 107007 |
| ALA | chi1 | -178 | 105977 |
| ALA | chi1 | -177 | 104639 |
| ALA | chi1 | -176 | 103918 |

# Here's the MDX Version

```
SELECT NON EMPTY {  [Structure].[Residue Hierarchy].[Residue Name]
                                    } on AXIS(0)
          , { CROSSJOIN ( { [Dihedral Angle].[Dihedral Hierarchy].[Angle Bin] }
                        ,  { [Measures].[Dihedral Count] } )}
    ON AXIS(1)
FROM [UnifiedDSV]
```

# MDX Results

- ~6 lines
- Returned the same results as SQL, but conveniently pivoted for comparison (6,138 rows)
- Execution time: 4 seconds

| | | Alanine | Arginine | Asparagine | Aspartic acid | Cysteine | Glutamine | Glutamic acid |
|---|---|---|---|---|---|---|---|---|
| -180 | Dihedral Count | 53596 | 22116 | 76832 | 95041 | 674 | 8626 | 26797 |
| -179 | Dihedral Count | 107007 | 44841 | 155320 | 190478 | 1477 | 18000 | 55579 |
| -178 | Dihedral Count | 105977 | 46318 | 154486 | 192171 | 1457 | 18379 | 57658 |
| -177 | Dihedral Count | 104639 | 47175 | 153191 | 193343 | 1484 | 19194 | 59307 |
| -176 | Dihedral Count | 103918 | 47725 | 152868 | 192268 | 1415 | 19518 | 60685 |
| -175 | Dihedral Count | 101764 | 47928 | 149364 | 190889 | 1396 | 19569 | 62164 |

# Contacts

- Atom-Atom contacts are frequently analyzed in simulations
- Two heavy atoms (i.e. not Hydrogen) are said to be in contact if they are less than 4.6 Å apart unless both atoms are Carbon; then they must be 5.4 Å apart or less

# Contact Matrices are BIG

- A brute-force comparison of all atoms in a simulation frame is the Cartesian product of all rows in that frame divided by two
- For 1enh, that amounts to 631,688 comparisons PER FRAME
- A SQL implementation involves a self-join on a Coordinate table

# SQL to just compute distances

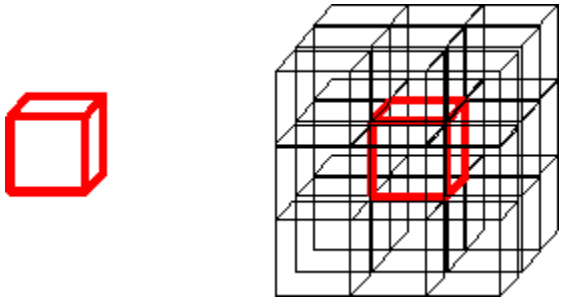
```
SELECT     c1.sim_id
        , c1.step
        , c1.struct_inst AS struct_inst1
        , c1.struct_id AS struct_id1
        , c1.atom_number AS atom_number1
        , c2.struct_inst AS struct_inst2
        , c2.struct_id AS struct_id2
        , c2.atom_number AS atom_number2
        , SQRT ( SQUARE( c1.x_coord - c2.x_coord )
                + SQUARE( c1.y_coord - c2.y_coord )
                + SQUARE( c1.z_coord - c2.z_coord )) AS [dist]
FROM dbo.Coord_112 AS c1
        JOIN dbo. Coord_112 AS c2
        ON ( c1.sim_id = c2.sim_id
            AND c1.step = c2.step
            AND (
                -- different instances
                ( c1.struct_inst <> c1.struct_inst )
                -- different atoms in same structure
                OR ( c1.struct_inst = c1.struct_inst
                    AND c1.atom_number <> c2.atom_number
                    AND c1.atom_number < c2.atom_number )))
```

# Brute Force SQL Result

- Limiting to heavy atoms, and applying filtering based on distances for a ***single*** 1enh simulation:
- Result: 36,210,336 rows, 2 hours 26 minutes
- Clearly not scalable…

# Hash3D Optimization

- For contact distances, we can safely exclude atoms more than 5.4Å apart
- Simulation box can be divided into 5.4Å cubes, each atom can be placed in a cube
- “bin” – a 1-dimensional integer hash can uniquely identify a cube
- “neighbors” are the 26 adjacent cubes



# Bins Stored with Coordinates

- Bins are computed and stored with each simulation at load time
- A C# Stored Procedure computes neighbors for each bins, and is stored in another table and indexed (under 1 second)
- Contact query with Hash3d: 36 minutes

| sim_id | struct_id | struct_inst | atom_number | step | x_coord | y_coord | z_coord | bin |
|---|---|---|---|---|---|---|---|---|
| 678 | 122 | 1 | 1 | 0 | -5.846 | 8.722 | 11.445 | 408 |
| 678 | 122 | 1 | 2 | 0 | -5.989 | 8.026 | 12.191 | 480 |
| 678 | 122 | 1 | 3 | 0 | -4.842 | 8.797 | 11.24 | 408 |
| 678 | 122 | 1 | 4 | 0 | -6.157 | 9.627 | 11.775 | 480 |
| 678 | 122 | 1 | 5 | 0 | -6.634 | 8.372 | 10.247 | 408 |

# MDX?

- Cube design is in progress
  - Building a dimension and hierarchy using bin and neighbors
  - Determining syntax to utilize hierarchy and find results
- A manuscript describing hash3d, support functions, tables, and index design and in progress

# CONCLUSIONS

# OLAP

**Good**

- Queries can be FAST
- Storage seems to be extremely efficient
- Certain classes of queries seem trivial to write (and much less complicated than SQL)

**Bad**

- MDX syntax can be complicated
- Shares keywords but no semantics with SQL
- Processing time and initial set up are non-trivial
- Documentation is often lacking sufficient detail

# Conclusions and Future Directions

- OLAP/MDX and SQL are complementary technologies, not replacements for each other
- More investigation is needed to tune OLAP design to maximize performance and usability
- Specific Next Steps
  - Finish hash3d OLAP design and compare to SQL
  - Additional performance and scale testing

# Acknowledgements


- Special thanks to Amanda Jonsson and Rudesh Toofanny for their insights and help

# Questions?

http://www.dynameomics.org