Object-oriented Model

- Managed system class
- Physical Equipment: superclass to "managed system"
- Managed system: contains objects of class "TCP connection"
- TCP Connection: maintains relationship to objects of class "endpoint"

Table-based Model

Two tables

- Managed system table: one entry (for the one deivce)
- TCP connections table: multiple entries (for multiple connections)
- A bit more "coarse" than the object-oriented model

ManagedSystem

SystemName	SystemContact	SystemLocation	SystemUptime

TCPConnections

LocalIP	LocalPort	RemoteIP	RemotePort	ConnectionState	PacketsSent	PacketsRecvd
...	...	...	...	...	...	...
...	...	...	...	...	...	...
...	...	...	...	...	...	...

Implicit Model based on Functions

Two groups of CLI functions

- 3 "system"-related functions: retreieve/set name, up time, contact, location
- 2 "connection"-related functions: retrieve connection information

Compared with object-oriented and table-based model

- The other two models: information itself instead of how to access infromation
- CLI-based implicit model: specific about how to access information.

MIB in SNMP

What does a MIB look like in practice?

Consider the specific metaschema and schema used in SNMP

- Metaschema: MOB specification language
- Schema: model

Metaschema:

- Structure of Management Information (SMI)
- Table-based

Schema:

- MIB-2
- For devices implementing TCP/IP, widely implemented

SMI and MIB-2 defined by the Internet Engineering Task Force (IETF)

- Standards documents published by IETF: Requests For Comments (RFC)
- RFC 793 - Transmimssion Control Protocol (TCP)
- RFC 1155 - SMI
- RFC 1213 - MIB-II

Minimize details of SMI and MIB-2

Our focus: insight into

- The level of information that is specified
- What a specification can look like

Structure of Managment Information (SMI)

SMI specifies MIB models in modules

Each MIB module serves a purpose

- Related to a device's communication interfaces
- Related to a feature embedded on a device

MIB of any managed device instantiates multiple MIB modules

- Each represents one aspect of the device & provides a structured view of the resources to be managed.

Object Identifier (OID) and OID Tree

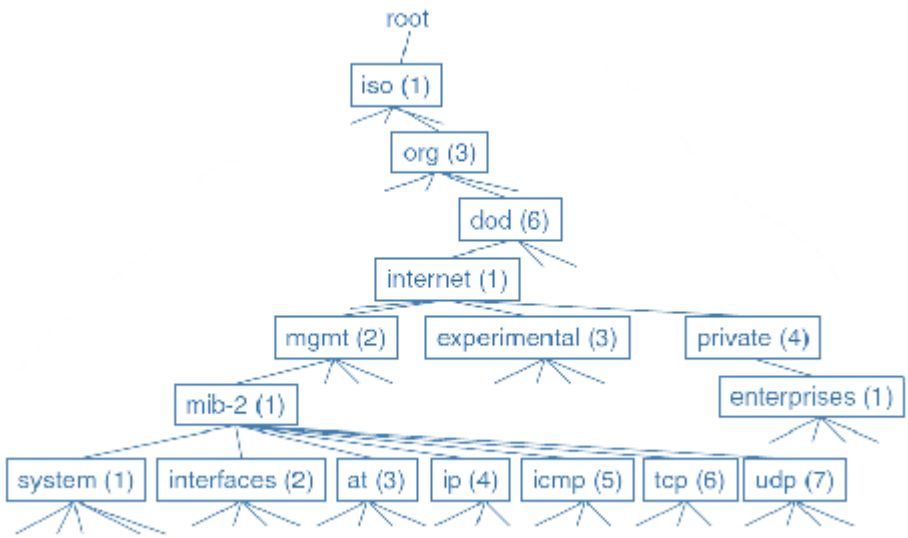
Infomration in MIB organized into a conceptual tree

- Every model in the MIB --> a node in the tree

Each node named relative to a containing node:

- Node ID: the object identifier (OID) that contains arbritary data
- Object type: Contains real information
- OID of iso: 1
- OID of org: 1.3
- OID of dod: 1.3.6
- OID of the internet: 1.3.6.1

- The tree - OID tree



Scalar and Tabular Objects

Another way to categorize: instantiated how many times in an agent?

Scalars: instantiated only *once* in an agent

- Nodes containing the host name, or a serial number of a chassis, or some global settings for the device,e tc.
- Management operations apply exclusively to scalar objects in SNMP

Tabular objects: instantiated multiple times

- Nodes representing infromation about cards in a chassis, or communication resources that are dynamically created and torn down during run time.
- Conceptual table entries for deleopers of management application to organize information in SNMP.

SMI-specified Data Types

No complex data types such as arrays or lists... There are only simple data types such as:

- Integer, octet strings, etc.
- Network addresses
- Counters - increasing only until reaching max value or resetting
- Gauges - increating or decreasing

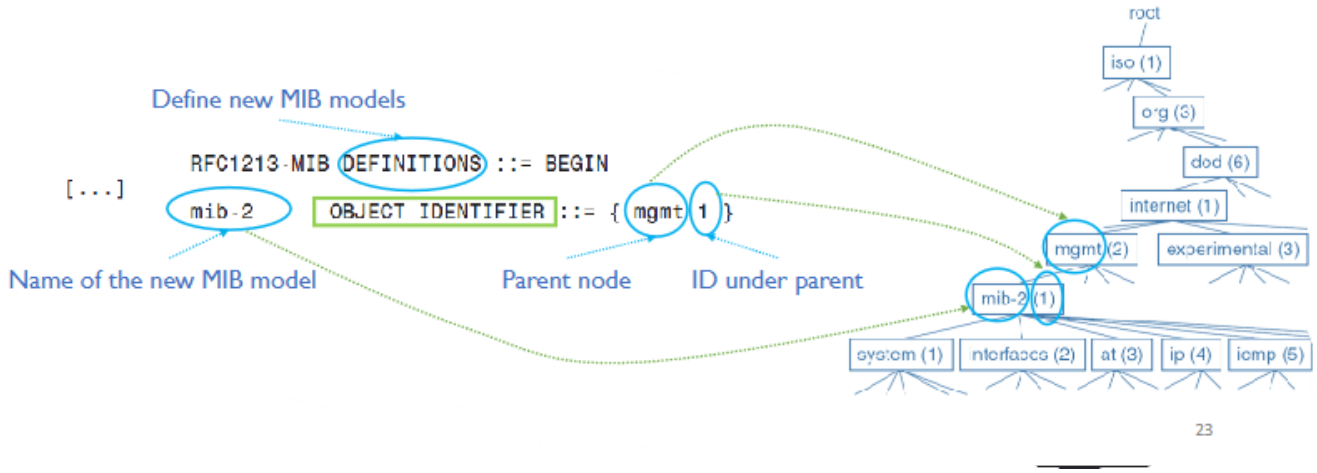
A conceptual structure can be built by grouping objects together.

A Closer Look at MIB-2

Defining the MIB-2 Module

Based on IETF RFC1213.

Establishes mib-2 as a new node under "mgmnt" in OID tree



Specify submodules (child nodes) of MIB-2

- Only 8 of 200+ child nodes shown below

-- groups in MIB-II

system	OBJECT IDENTIFIER ::= { mib-2 1 }	System group
interfaces	OBJECT IDENTIFIER ::= { mib-2 2 }	Interface group
at	OBJECT IDENTIFIER ::= { mib-2 3 }	Address translation
ip	OBJECT IDENTIFIER ::= { mib-2 4 }	Internet protocol
icmp	OBJECT IDENTIFIER ::= { mib-2 5 }	Internet control message protocol
tcp	OBJECT IDENTIFIER ::= { mib-2 6 }	Transmission control protocol
udp	OBJECT IDENTIFIER ::= { mib-2 7 }	User datagram protocol
egp	OBJECT IDENTIFIER ::= { mib-2 8 }	Exterior gateway protocol

---

Define A Submodule of MIB-2

- Elements of the definition of an object type
- *Syntax*: Data type (e.g "string" in sysDescr)
- *Access*: read-write (if can be set by a manager), read-only, or not-accessible
- *Status*: definition lifecycle (mandatory, optional, obsolete)
  - mandatory: every implementation of the module must include it
- *Description*: explanation
  - Intended purpose of the object type
  - Aspects to be implemented

Define A (Conceptual) Table

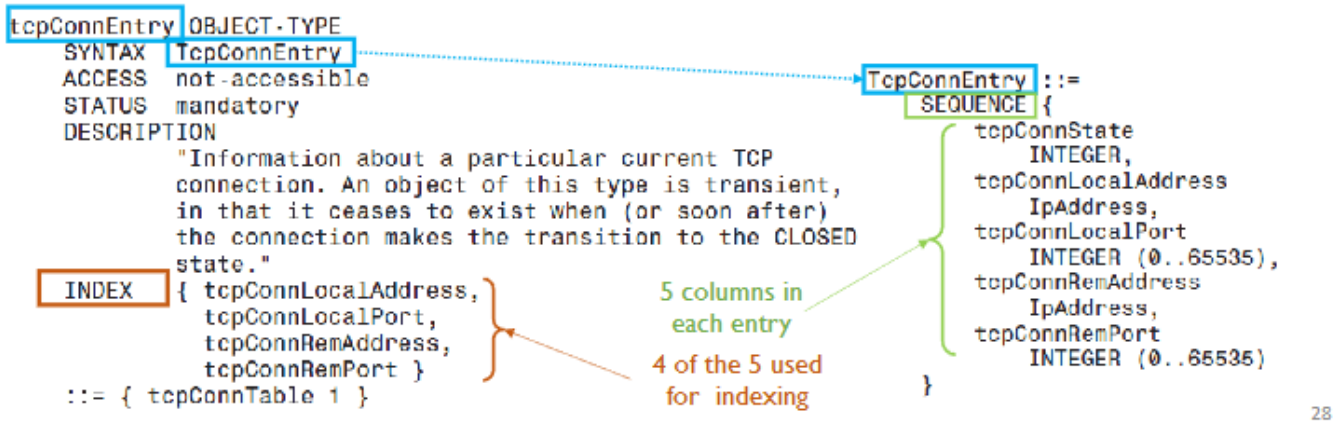
- *TCP* (OID 1.3.6.1.2.1.6) has 20 child nodes
  - Some includes the *definition of a table*
- *Syntax* (data type): *SEQUENCE OF* another object
  - tcpConnTable is a table, each row is a tcpConnEntry object
- *Access*: tcpConnTable not accessible
- Carries no info on its own; just a container object for objects carrying info.

OID	Name	Sub children
1.3.6.1.2.1.6.1	tcpRtoAlgorithm	0
1.3.6.1.2.1.6.2	tcpRtoMin	0
1.3.6.1.2.1.6.3	tcpRtoMax	0
⋮		
1.3.6.1.2.1.6.13	tcpConnTable	1
1.3.6.1.2.1.6.14	tcpInErrs	0
1.3.6.1.2.1.6.15	tcpOutRsts	0
1.3.6.1.2.1.6.16	ipv6TcpConnTable	2
1.3.6.1.2.1.6.17	tcpHCInSegs	1
1.3.6.1.2.1.6.18	tcpHCOutSegs	1
1.3.6.1.2.1.6.19	tcpConnectionTable	1
1.3.6.1.2.1.6.20	tcpListenerTable	1

27

Table Entry

- "tcpConnEntry" - a conceptual row/entry of "tcpConnTable"
  - Syntax: not a simple data type but a sequence
  - *t*cpConnEntry object and *T*cpConnEntry syntax are different
  - Index: unique to table entry objects (specifying the elements used as the index)



Columnar Object

A table (and each entry of the table) has a set of columns

- Each column corresponds to a columnar object.

Not-Accessible vs Accessible

- tcpConnTable not accesible - just a conceptual container for entries
- tcpConnEntry also not accesible - a conceptual container of elements
- *Columnar objects: accessible - elements of the array*

Instances and Instance OID

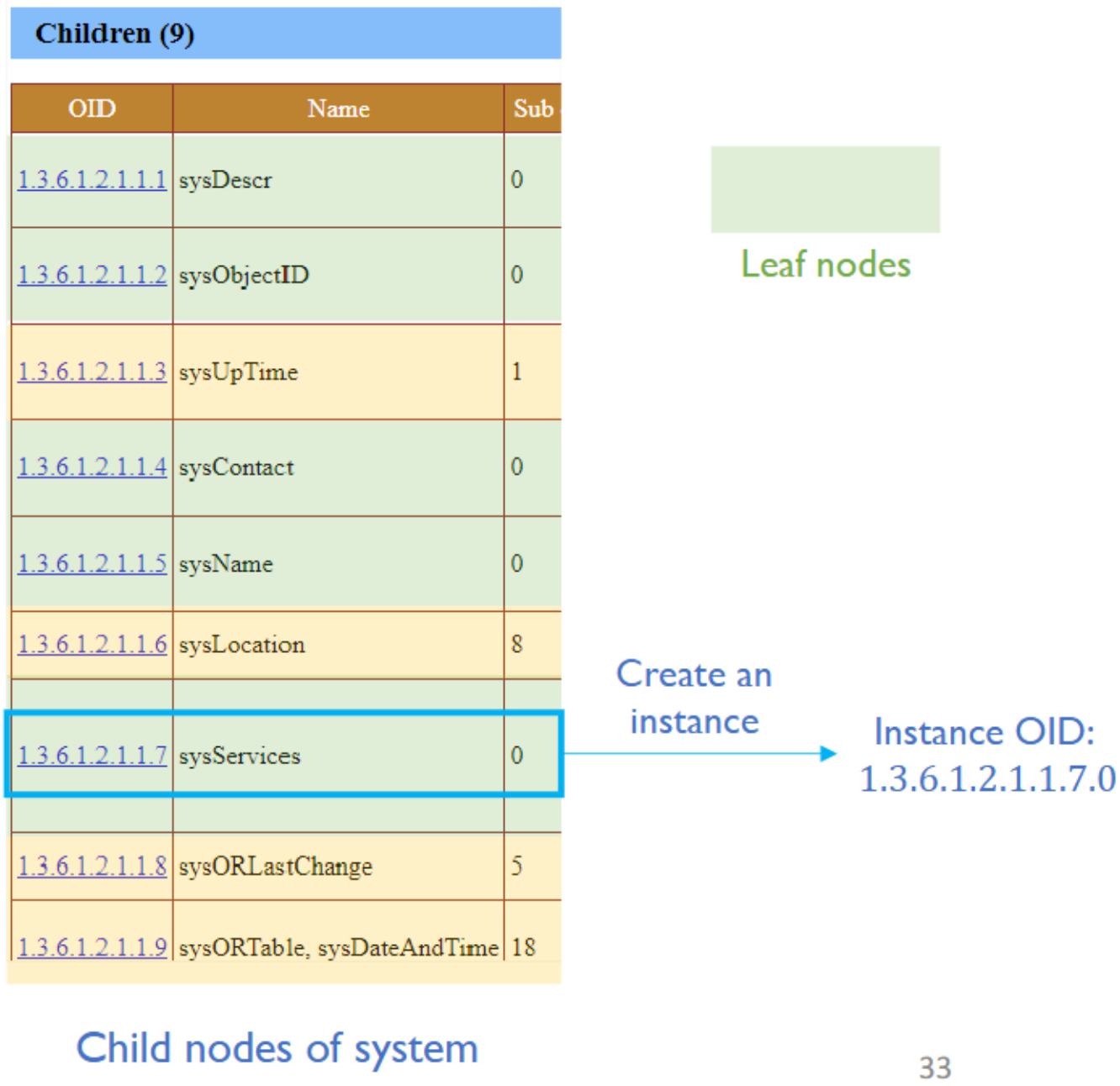
Instantiation in SNMP MIB

- Instances of an object type contain the actual values that a manager can retrieve from the agent.
- In SNMP, instances are also considered as part of the OID tree
- *Only leaf nodes can be instantiated*
    - Leaf nodes grow new leaf nodes (instances)

Instance OID

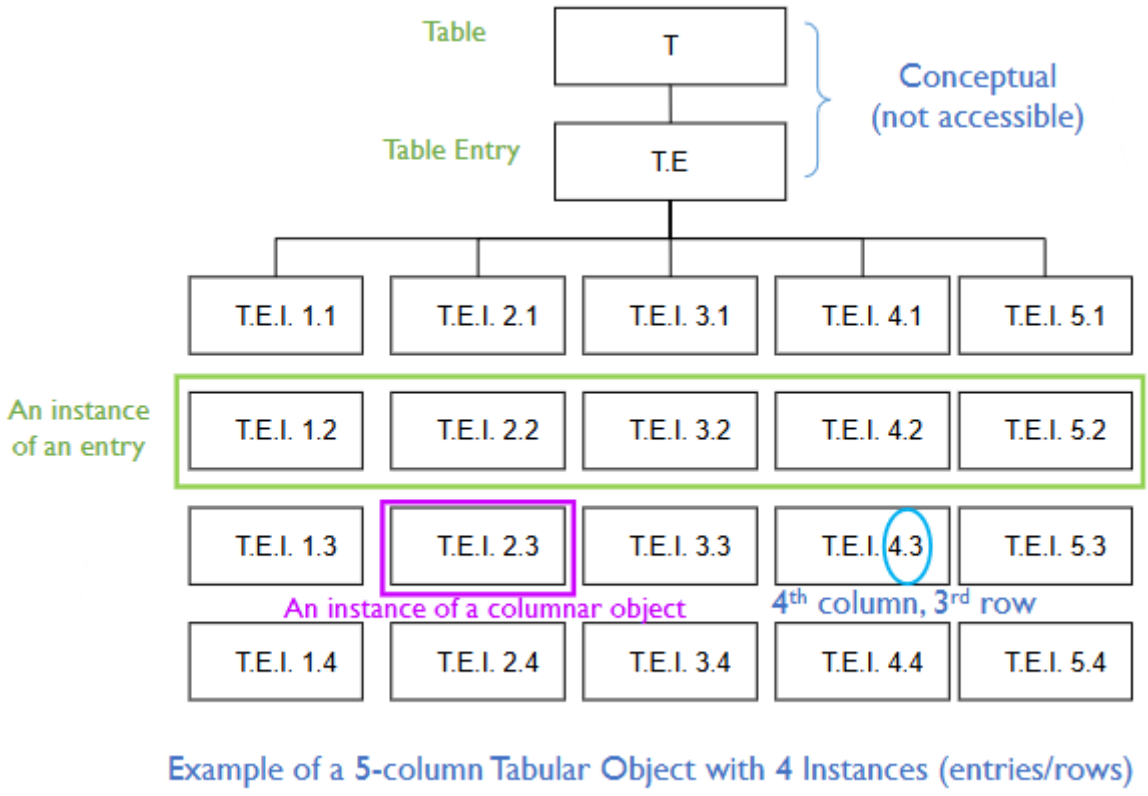
Scalar Objects:

- Scalar object can have only one instance
    - Object instance has an OID too
- OID of the object instance = OID of the object + suffix (.0)
- Consider an instance of object sysServices (1.3.6.1.2.1.1.7)
    - Instance OID: 1.3.6.1.2.1.7.7.0



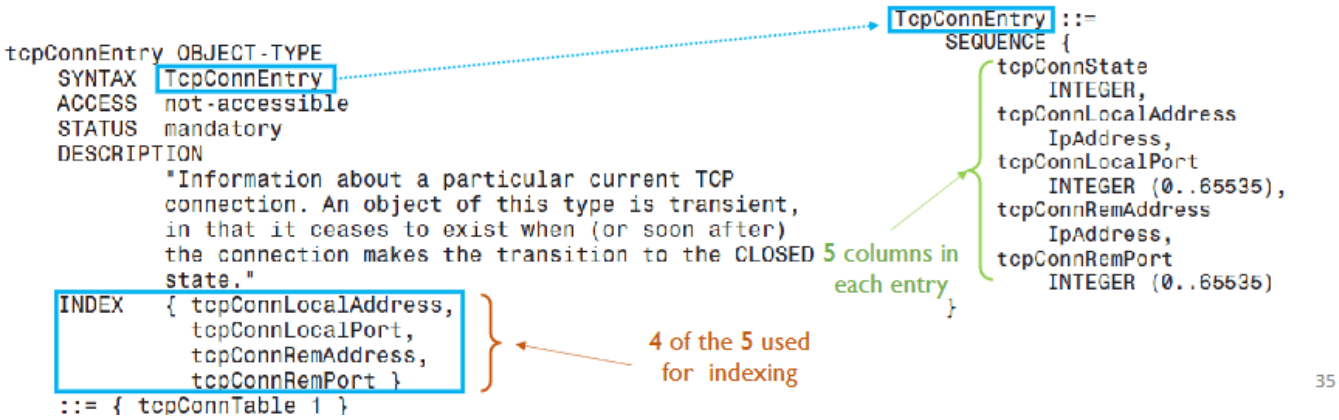
Columnar Objects:

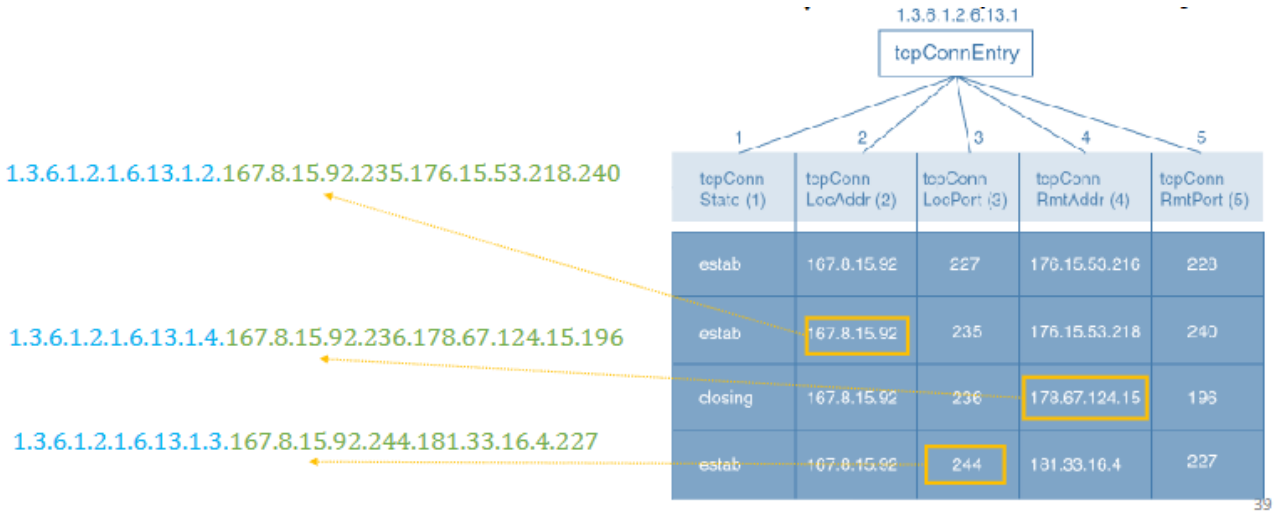
- A table can have multiple entries (multiple instances of the table entry)
  - Each entry has multiple columnar objects
  - Each columnar object needs to be distinguished



Example of a 5-column Tabular Object with 4 Instances (entries/rows)

- Recall that table entries have an "index" field in its definition
  - Index field specifies the column(s) used for indexing
  - *Index* --> Used to help identify each instance of a columnar object
- OID of a columnar object instance = OID of the object + index value





3 instances shown in photo for tcpConnEntry:

First: 1.3.6.1.2.1.6.13.1.2. *167.8.115.92* .235.176.15.53.218.240

- This presents the instance of tcpConnEntry local address. Which is *167.8.15.92*

Second: 1.3.6.1.2.1.6.13.1.4.167.8.15.92.236.* 178.67.124.15* .196

- This presents the tcpConn RmtAdd instance. Which is 172.67.124.15

Third: 1.3.6.1.2.1.6.13.1.3.167.8.15.92. *244* .181.33.16.4.227

- This would give the instance of tcpConnLocPort 244.

In each part I highlighted how the relevant information is found based on the explanation given above.

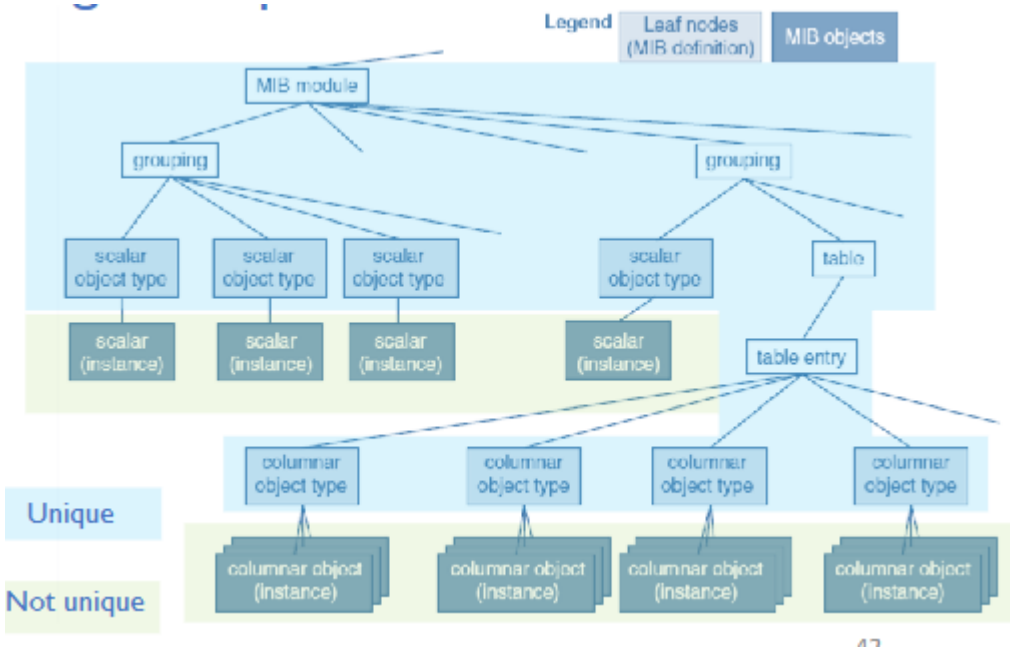
> - Recall that table entries have an "index" field in its definition
> - Index field specifies the column(s) used for indexing
> - *Index* --> Used to help identify each instance of a columnar object
> - OID of a columnar object instance = OID of the object + index value
>
> Make sure to know what the index columns are because that's all you will need to enter.

Uniqueness of OID

- OIDs of MIB models are globally unique.
- OIDs of object instances are no longer unique

Example:

- Two routers both implement MIB-2
- Both have an object instance with OID 1.3.6.1.2.1.1.3.0 - the instance of the scalar object sysUpTime
- The system uptime of the two routers are probably different.



Modeling Management Information

Management Information: Abstraction

- Companies can develop proprietary MIB models
- Management information that an agent exposes across its management interface: an *abstraction* of the managed device
  - The abstraction - based on MIB models
  - Information in the MIB - instantiations of these models

Relevant & Irrelevant Information

Relevant information to management - include in abstraction:

- Device's serial number
- Software version running on device
- Timeout values for a particular protocol

Irrelevant infromation to management - exclude from abstraction:

- Color of the chassis
- number of chips on the main board
- size of last-transmitted packet

Proper Abstraction

- Finding the proper abstraction is not always easy
- Not always obvious what information will be needed.

    - Includes the time at which the last critical alarm occured or not?
    - Keep packet counter statistics on each type of packet or just the sum

Q: What would happen if we include too little management information?

> - Some management decisions must be made with limited information
> - Fewer posibilities to fine-tune network performance because certain settings cannot be configured ("write" access)

Usually suggested to include a bit more than the minimum required.

Q: What would happen if we include too much management information in the abstraction

> - Management interface can be more complex than necessary
> - Users need to learn to interpret more pieces of management information
> - Memory footprint of the management agent in the device increases
> - More effort in management application development

- Too much information: too many management knobs and displays
    - Model developers need to know the purpose of the management information
    - Do not include a real-world aspect of a device just because it is there

Finding balance is a matter of design. And to do so, these questions would be good to ask:

```
1. Why a piece of management information might be useful?

2. Are there enough displays that tell network managers what is going on at the
managed device?

3. Are enough "knobs" provided to configure the device, to provision services over
it, and to tune network performance?

4. Can all the relevant management scenarios for the varios management functions
can be supported with the provided management infromation?

5. What is the proper granularity of the model?

6. Can the model be easily extended?
```