Oracle Developer Essentials: Tables and Indexes

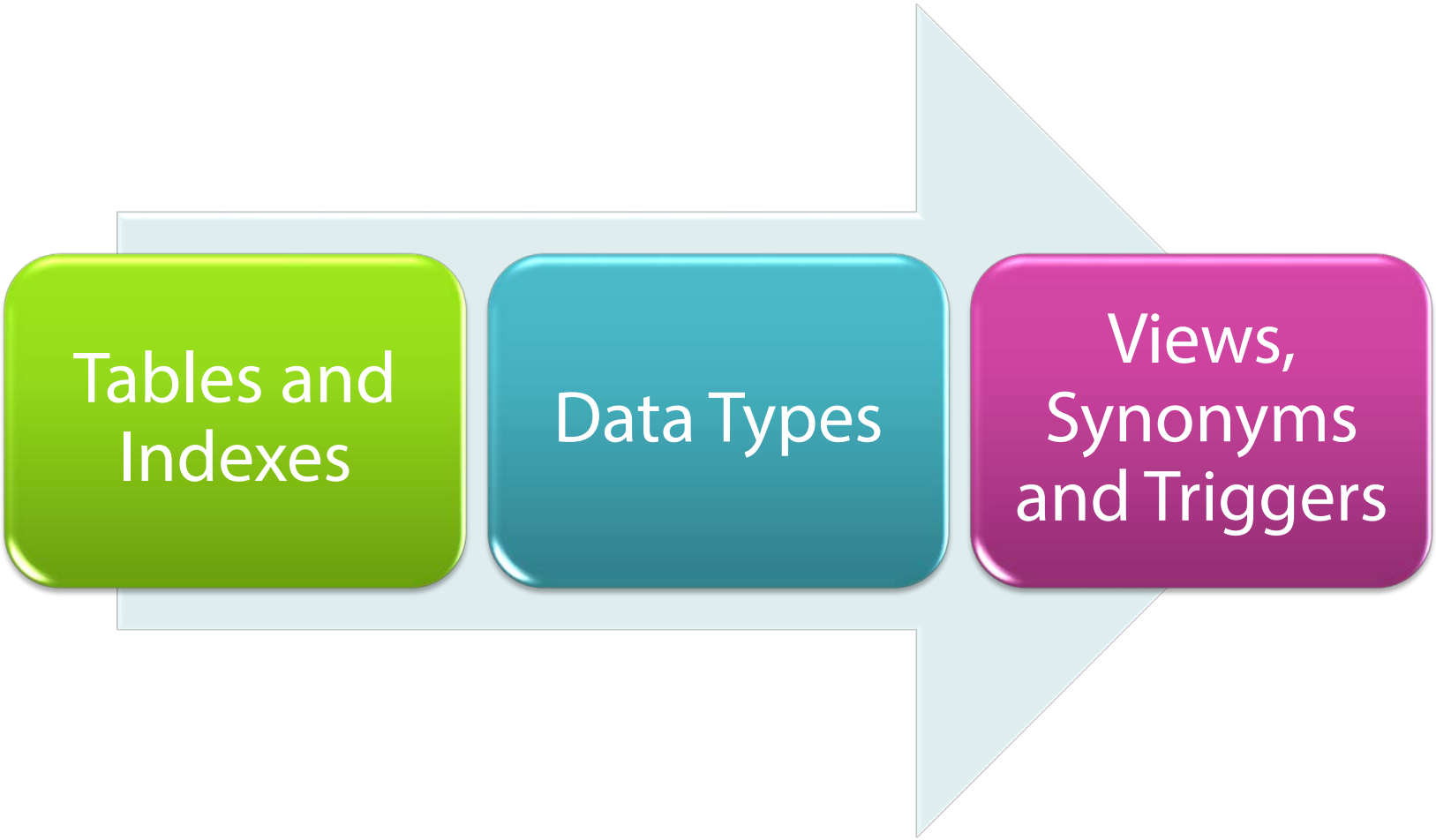
Introduction

David Berry
http://buildingbettersoftware.blogspot.com/

pluralsight
hardcore dev and IT training

Course Outline

Oracle Essentials Course Sequence



Database Lifetime and Usage

Used by **multiple** applications

- Most data has multiple **consumers**
- **Many applications** will be connected over the **lifetime** of the database

Used for **long periods** of time

- Often used for **five** or more years
- **Historical** data is often important

Motivation

Understand all the available options

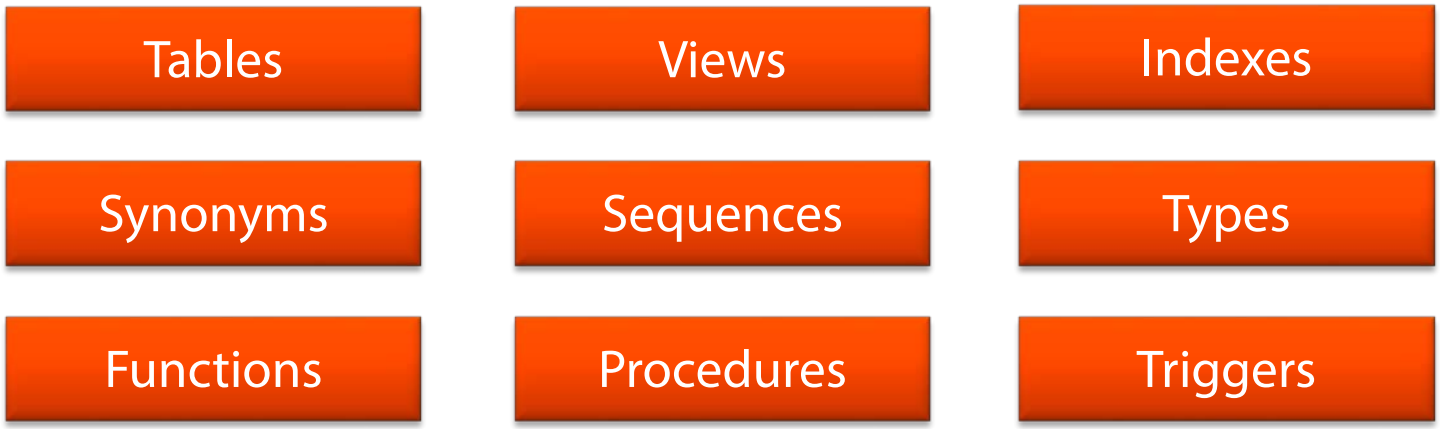
Build databases that are easy to use

Build databases that perform well

Terminology

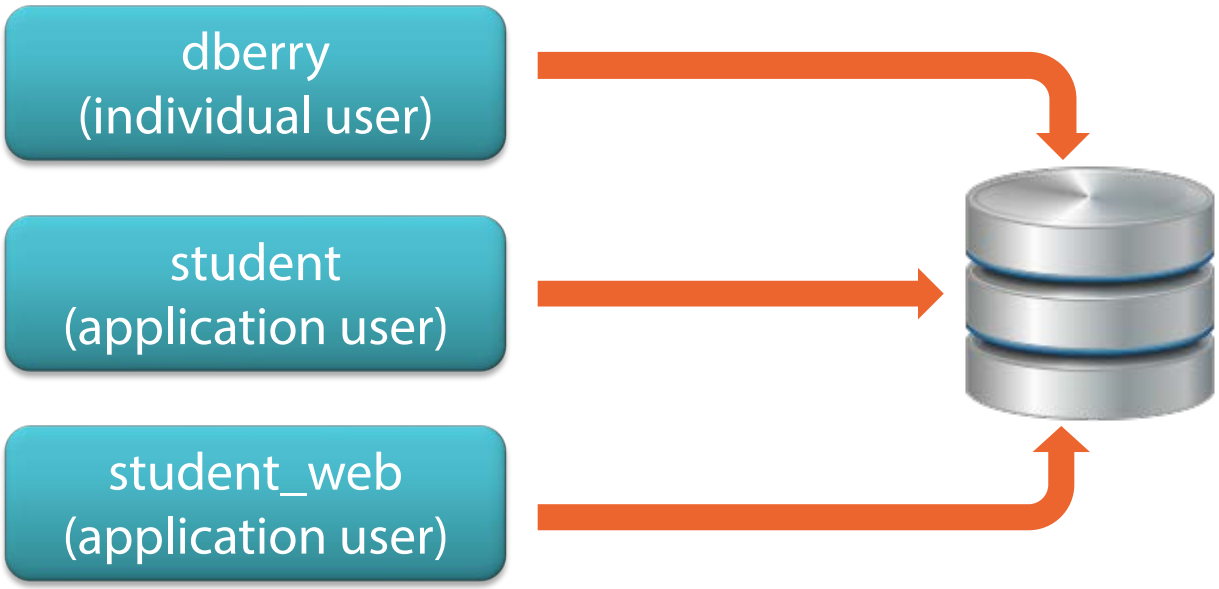
Database Objects

Generic term for a structure owned by a user in an Oracle database



Users in Oracle

User	A user within the Oracle database



Schemas

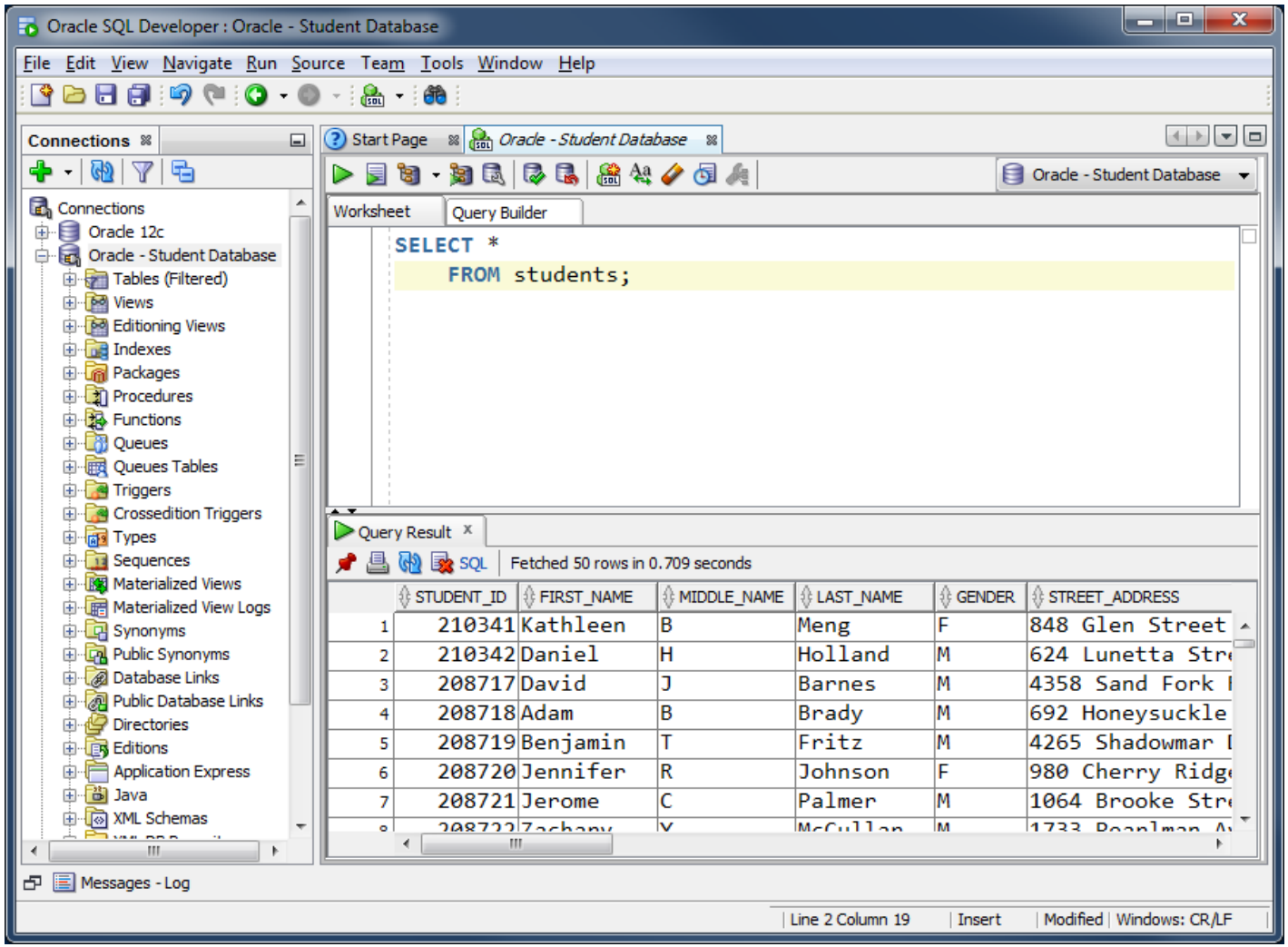
Definition	All of the objects that are owned by a user
As a Container	Often correlate to an application or business function

Accessing Objects in a Different Schema

```
SELECT
    department_code, course_number, name, description
FROM student.courses
```

Schema name must be included when accessing the table as a different user

Oracle SQL Developer

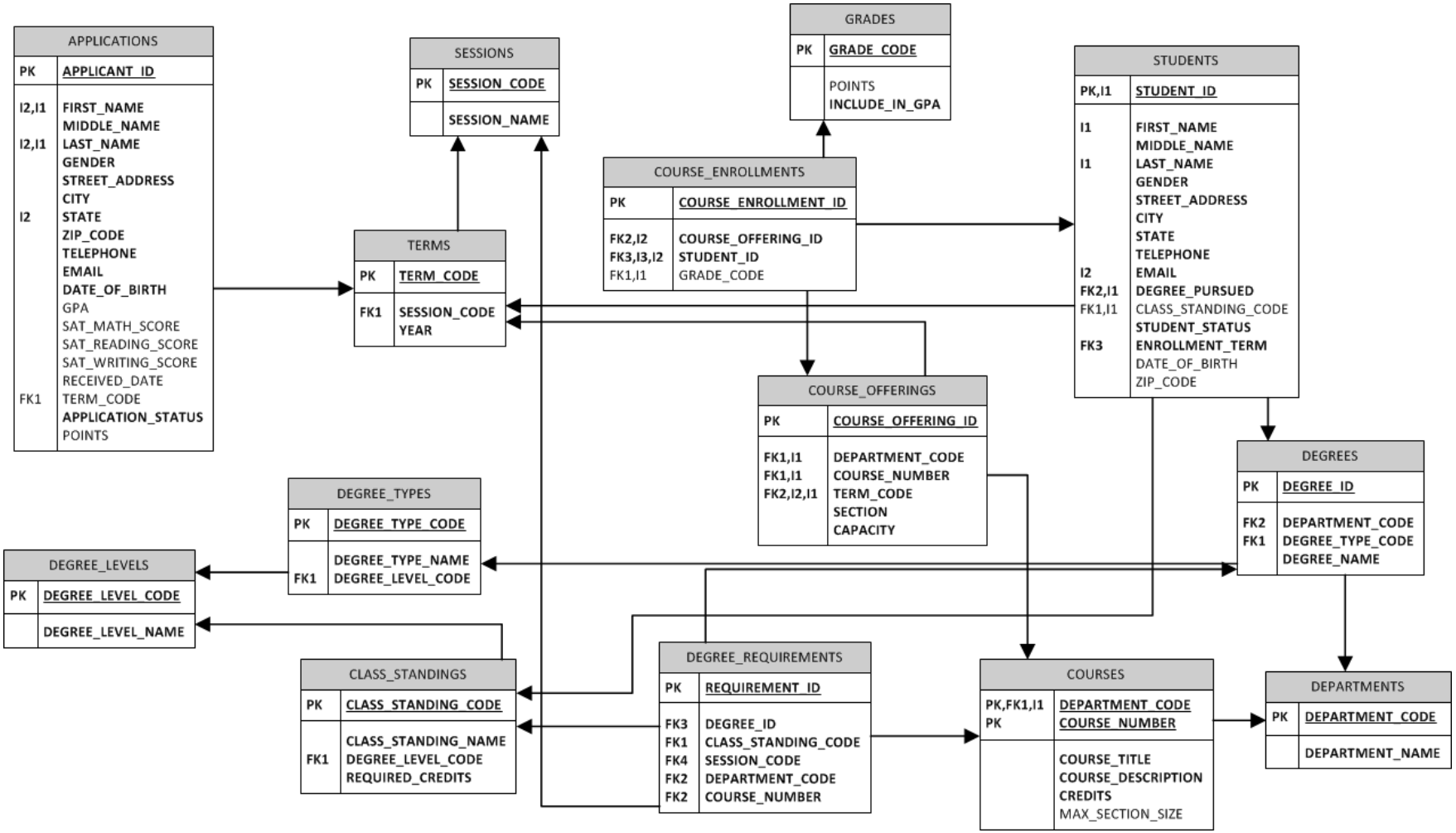


http://bit.ly/OracleSqlDeveloper

Course Conventions

- Course will use SQL statements to demonstrate capabilities
- SQL syntax better communicates different options
- Can easily to transition to graphical tools later
- Syntax of commands important for reading scripts

Course Resources



http://bit.ly/OracleCourseSchema

Oracle Resources

Oracle 12c Documentation Home

http://bit.ly/Ora12cDocs

Oracle Developer Virtual Machines

http://bit.ly/OracleVms

Summary

Course overview and motivation

Common Oracle terms

Oracle resources

Table Basics

David Berry
http://buildingbettersoftware.blogspot.com/

Introduction to Tables

student_id	first_name	last_name	phone	email	major_id
208718	Adam	Brady	360-592-7886	AdamBBrady@dayrep.com	3
208723	George	Ward	216-952-0212	GKWard@superrito.com	5
208728	Marcia	Garcia	630-239-4325	MarciaDGarcia@rhyta.com	8
208741	Walter	Stumpf	419-500-5489	WalterCStumpf@einrot.com	12
208749	Brenda	Brown	979-859-7105	BrendaBrown@dayrep.com	5
208755	Christopher	Glenn	803-980-0813	ChrisGlenn@superrito.com	5
209628	Lisa	Shifflett	229-409-0015	lshifflett@jourrapide.com	4

Tables Roadmap

Table Basics
(this module)

- Create table syntax
- Naming rules
- Null values
- Default values

Table Constraints
(next module)

- Primary keys
- Foreign keys
- Check constraints

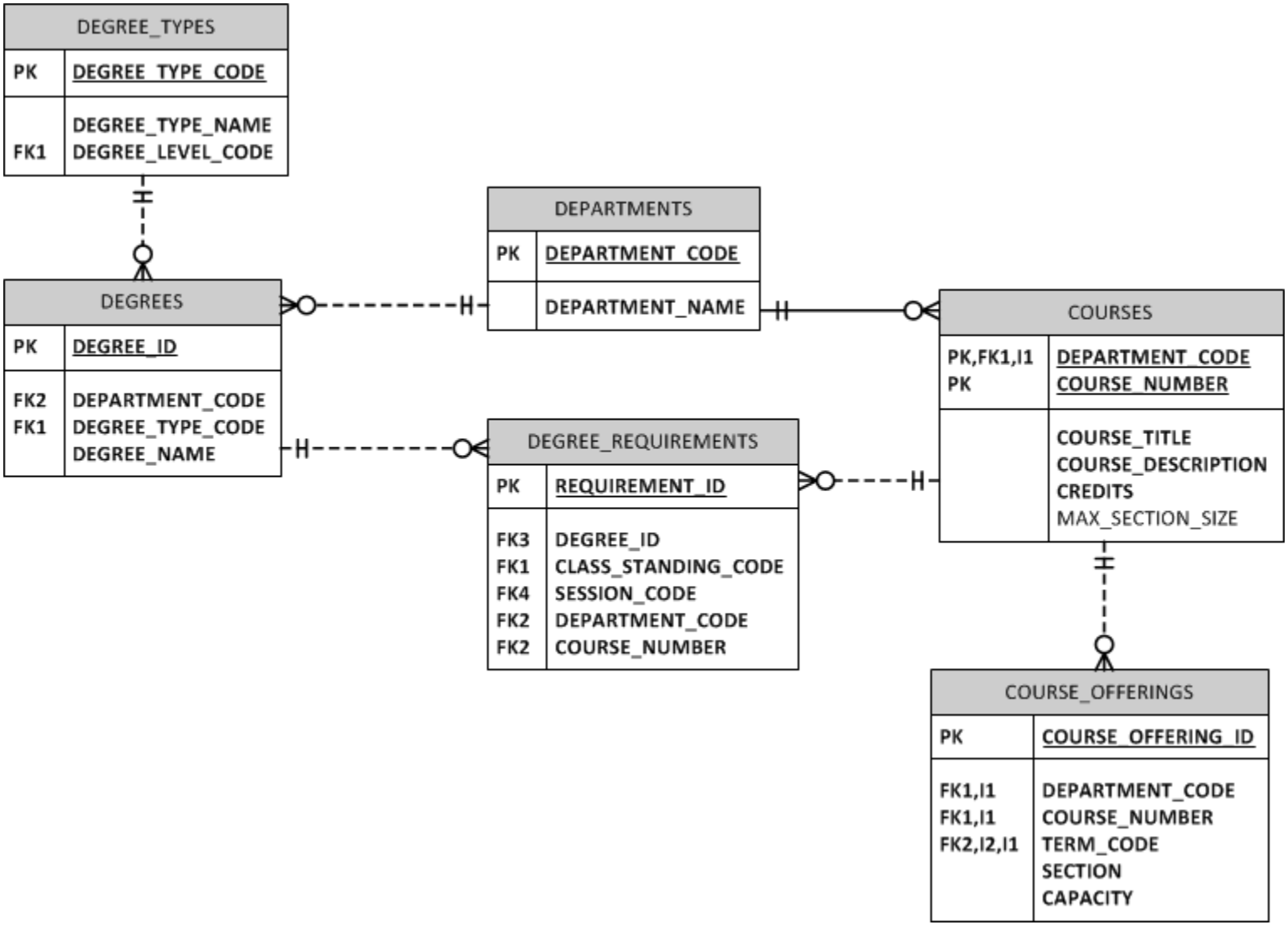
Table Storage
(in two modules)

- Storage characteristics
- Storage options

Naming Rules

Maximum Length	30 characters for all objects
Uniqueness	Tables, views must be unique within schema Columns must be unique within a table
SQL Reserved Words	Possible to use but discouraged

Tables Represent Business Data



Basic Create Table Syntax

```
CREATE TABLE courses
(
    department_code     VARCHAR2(2)     NOT NULL,
    course_number       NUMBER(3,0)     NOT NULL,
    course_title        VARCHAR2(64)    NOT NULL,
    course_description  VARCHAR2(512)   NOT NULL,
    credits             NUMBER(3,1)     NOT NULL,
    CONSTRAINT pk_courses PRIMARY KEY
        (department_code, course_number),
    CONSTRAINT fk_courses_department_code FOREIGN KEY
        (department_code) REFERENCES departments (department_code)
    CONSTRAINT ck_courses_course_number
        CHECK (course_number BETWEEN 100 AND 999)
)
TABLESPACE users
PCTFREE 75;
```

Naming Tables and Columns

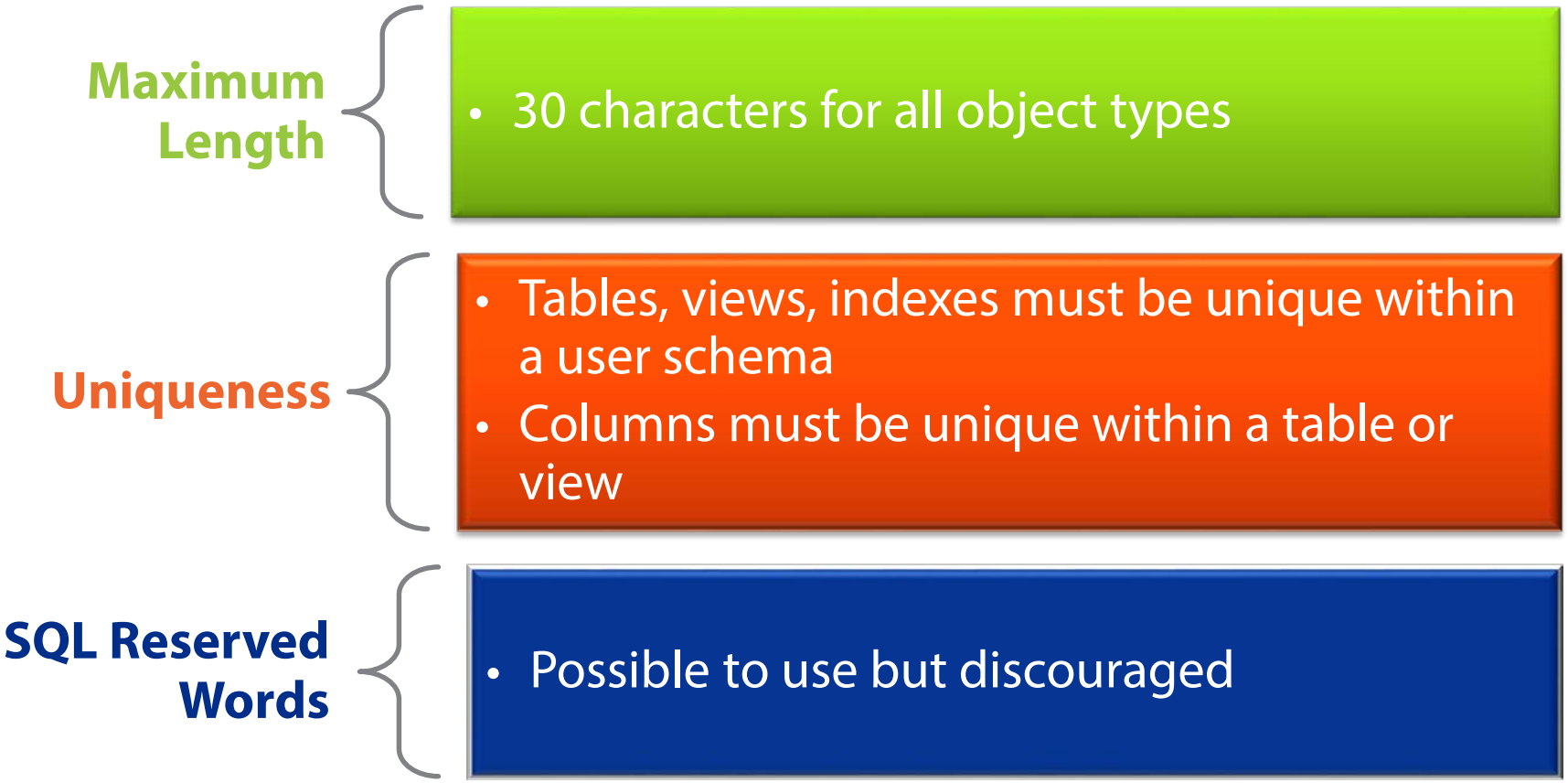
Good names communicate intent

Poor names lead to confusion

Naming Rules

Maximum Length	30 characters for all objects
Uniqueness	Tables, views must be unique within schema Columns must be unique within a table
SQL Reserved Words	Possible to use but discouraged

Naming Rules



Unquoted Names

- **Most common way of naming objects**

```
CREATE TABLE zip_codes
(
    zip_code      VARCHAR2(5)       NOT NULL,
    city          VARCHAR2(30)      NOT NULL,
    state         VARCHAR2(30)      NOT NULL
);
```

- **Allowable characters are**
  - Alphanumeric characters
  - Underscore (_) character
  - $ and # characters

- **Names will be case insensitive**

Unquoted Names Are Case Insensitive

```
CREATE TABLE zip_codes
(
    zip_code      VARCHAR2(5)       NOT NULL,
    city          VARCHAR2(30)      NOT NULL,
    state         VARCHAR2(30)      NOT NULL
);
```

Allowed SQL Statements

```
-- This is OK
select ZIP_CODE, CITY, STATE
    from ZIP_CODES;

-- So is this
SELECT Zip_Code, City, State
    FROM Zip_Codes;
```

Quoted Identifiers

- **Allows a wider range of options**

```
CREATE TABLE “ZipCodes”
(
    “zip code”        VARCHAR2(5)        NOT NULL,
    “city.name”       VARCHAR2(30)       NOT NULL,
    “state-abbr”      VARCHAR2(30)       NOT NULL
);
```

- **You must also use quoted names in SQL statements**

```
SELECT
    “zip code”, “city.name”, “state-abbr”
FROM “ZipCodes”
WHERE “zip code” = ‘54911’
```

Defining Columns

```
CREATE TABLE students
(
    student_id        NUMBER(6)       NOT NULL,
    first_name        VARCHAR2(30)    NOT NULL,
    last_name         VARCHAR2(30)    NOT NULL,
    email_address     VARCHAR2(128)   NOT NULL,
    phone             VARCHAR2(20)    NULL,
    city              VARCHAR2(30)    NULL,
    state             VARCHAR2(2)     NULL,
    status_code       VARCHAR2(1)     DEFAULT ‘A’ NOT NULL,
    CONSTRAINT pk_students PRIMARY KEY (student_id)
);
```

Column Definition Rules

Number of Columns per Table

- Maximum of 1000 columns
- Over 255 columns results in row chaining

Column Naming Rules

- 30 characters maximum
- Names must be unique within a table
- Names can be reused in different tables

Minimum Information

- Name and datatype must be specified
- NULL an default clauses are optional

NULL Column Values

```
CREATE TABLE students
(
    student_id     NUMBER(6,0)     NOT NULL,
    first_name     VARCHAR2(30)    NOT NULL,
    last_name      VARCHAR2(30)    NOT NULL,
    city           VARCHAR2(30)    NULL,
    state          VARCHAR2(2)     NULL,
    date_of_birth  DATE
);
```

- **In character based columns (CHAR, VARCHAR, NCHAR, NVARCHAR)**
  - Empty string is treated as a NULL value

Default Values

```
CREATE TABLE students
(
    student_id          NUMBER(6,0)     NOT NULL,
    first_name          VARCHAR2(30)    NOT NULL,
    last_name           VARCHAR2(30)    NOT NULL,
    city                VARCHAR2(30)    NULL,
    state               VARCHAR2(2)     NULL,
    date_of_birth       DATE,
    enrollment_status   VARCHAR2(1)    DEFAULT ‘E’  NOT NULL
);
```

NULL, NOT NULL or DEFAULT VALUE

That is the question

Data You Do Not Have

Using NULL values

user_id	email_address	first_name	last_name	phone_number	city
8385930	AdamBBrady@dayrep.com	Adam	Brady	303-555-1212	Denver
8385931	GKWard@superrito.com	George	Ward	<NULL>	Phoenix
8385932	MarciaDGarcia@rhyta.com	Marcia	Garcia	<NULL>	<NULL>
8385933	WalterCStumpf@einrot.com	Walter	Stumpf	414-555-8965	<NULL>

Using default values

user_id	email_address	first_name	last_name	phone_number	city
8385930	AdamBBrady@dayrep.com	Adam	Brady	303-555-1212	Denver
8385931	GKWard@superrito.com	George	Ward	000-000-0000	Phoenix
8385932	MarciaDGarcia@rhyta.com	Marcia	Garcia	000-000-0000	UNKNOWN
8385933	WalterCStumpf@einrot.com	Walter	Stumpf	414-555-8965	UNKNOWN

An Event That Has Not Occurred

Using NULL values

order_id	customer_id	order_status_code	order_date	ship_date
567893	832	S	2014-08-03	2014-08-05
56784	944	S	2014-08-03	2014-08-07
567895	968	P	2014-08-05	<NULL>
567896	1024	N	2014-08-06	<NULL>

Using default values

order_id	customer_id	order_status_code	order_date	ship_date
567893	832	S	2014-08-03	2014-08-05
56784	944	S	2014-08-03	2014-08-07
567895	968	P	2014-08-05	1900-01-01
567896	1024	N	2014-08-06	1900-01-01

Use Defaults To Indicate an Initial State

Using default values

member_id	email_address	first_name	last_name	phone_number	membership_level
8385930	AdamBBrady@dayrep.com	Adam	Brady	303-555-1212	P
8385931	GKWard@superrito.com	George	Ward	602-555-3547	F
8385932	MarciaDGarcia@rhyta.com	Marcia	Garcia	608-555-2575	F
8385933	WalterCStumpf@einrot.com	Walter	Stumpf	414-555-8965	P

Using NULL values

member_id	email_address	first_name	last_name	phone_number	membership_level
8385930	AdamBBrady@dayrep.com	Adam	Brady	303-555-1212	P
8385931	GKWard@superrito.com	George	Ward	602-555-3547	
8385932	MarciaDGarcia@rhyta.com	Marcia	Garcia	608-555-2575	
8385933	WalterCStumpf@einrot.com	Walter	Stumpf	414-555-8965	P

NULL
versus
Default Values

Know your data

- **What does the data mean**
- **How is it used**

What communicates intent most clearly?

- **Would another user draw the same conclusion?**

Virtual Columns Introduction

Normal column

Data is **physically** stored on disk by Oracle

Virtual column

Value is **computed** from other columns in table

Virtual Column Use Case

```
CREATE TABLE orders
(
    order_id        NUMBER(10)    NOT NULL,
    order_date      DATE          NOT NULL,
    customer_id     NUMBER(10)    NOT NULL,
    subtotal        NUMBER(10,2),
    tax             NUMBER(10,2),
    shipping        NUMBER(10,2),
    grand_total     NUMBER(10,2)
);
```

Defining a Virtual Column

```
CREATE TABLE orders
(
    order_id        NUMBER(10)    NOT NULL,
    order_date      DATE          NOT NULL,
    customer_id     NUMBER(10)    NOT NULL,
    subtotal        NUMBER(10,2),
    tax             NUMBER(10,2),
    shipping        NUMBER(10,2),
    grand_total     NUMBER(10,2)
                        AS (subtotal + tax + shipping) VIRTUAL
);
```

Defining a Virtual Column With a Function

```
CREATE TABLE students5
(
    student_id     NUMBER(10)    NOT NULL,
    first_name     VARCHAR2(30)  NOT NULL,
    middle_name    VARCHAR2(30)  NOT NULL,
    last_name      VARCHAR2(30)  NOT NULL,
    email_address  VARCHAR2(60)  NOT NULL,
    email_domain   VARCHAR2(60)  AS (
        SUBSTR(email_address, INSTR(email_address, '@', 1, 1)+1)
        ) VIRTUAL
);
```

Virtual Column Properties

- Value is computed in result set of query
- Cannot INSERT into or UPDATE virtual columns
- Can only make use of columns in the same table
- Indexes can be created over virtual column values

Virtual Columns Summary

Useful for situations where a derived value is needed

Summary

CREATE TABLE Syntax

Naming Rules

NULLS and Default Values

Table Constraints

David Berry
http://buildingbettersoftware.blogspot.com/

Table Constraints Overview

- **Table constraints**
  - Define rules your data must follow

- **Examples**
  - Every student must have a unique student id number
  - Every course must be associated with a department

Module Outline

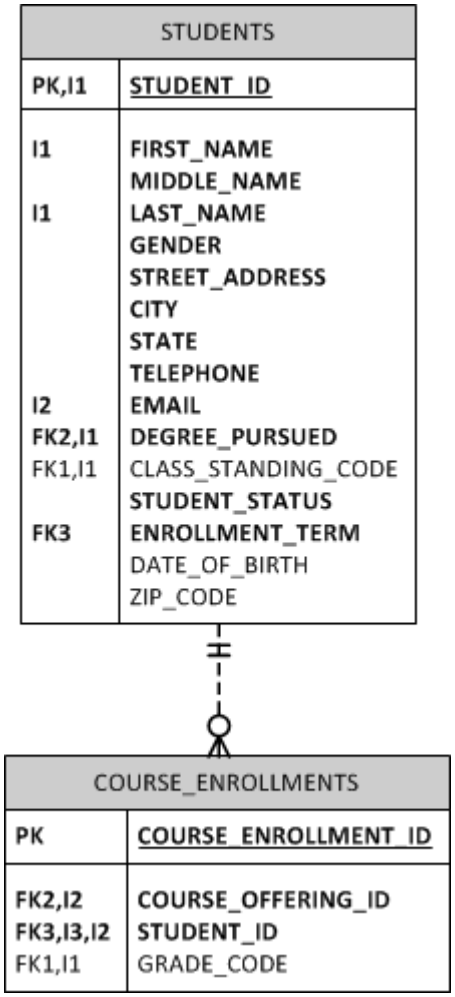
Primary Keys

Foreign Keys

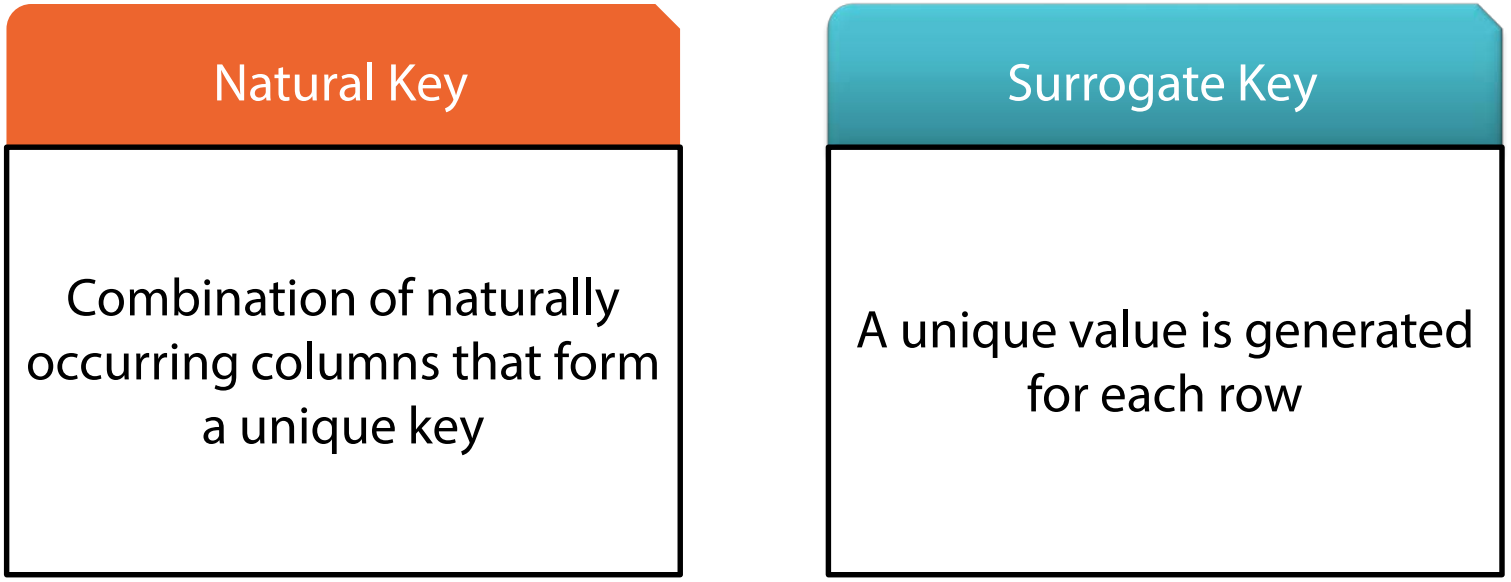
Check Constraints

Primary Keys

- Uniquely identifies each row in a table
- Can be defined over one or more columns
- Cannot contain a NULL value in any primary key column



Primary Key Types



The real reason I favor surrogate keys….

I am a terrible typist!

Primary Key Definition

```
CREATE TABLE departments
(
    department_code     VARCHAR2(2)     NOT NULL    PRIMARY KEY,
    department_name     VARCHAR2(64)    NOT NULL
);
```

Primary Key Using Constraint Clause

```
CREATE TABLE courses
(
    department_code     VARCHAR2(2)     NOT NULL,
    course_number       NUMBER(3,0)     NOT NULL,
    course_title        VARCHAR2(64)    NOT NULL,
    course_description  VARCHAR2(512)   NOT NULL,
    credits             NUMBER(3,1)     NOT NULL,
    CONSTRAINT pk_courses
        PRIMARY KEY (department_code, course_number)
);
```

Add a Primary Key to an Existing Table

```
ALTER TABLE courses
    CONSTRAINT pk_courses
    ADD PRIMARY KEY (department_code, course_number);
```

Primary Key Recommendations

Make sure **every table** has a **primary key** defined

Values of a **primary key** should be **immutable**

Oracle ROWID

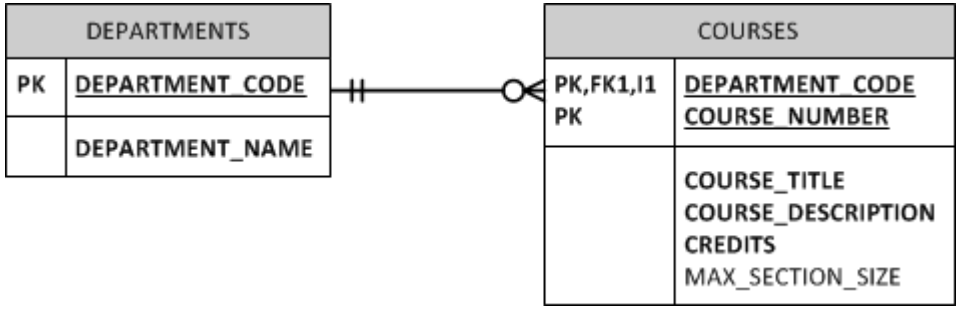
- **Represents the address of the row in the table**

```
SQL> SELECT rowid, zip_code, city, state
  2      FROM zip_codes;

ROWID              ZIP_C CITY                           ST
------------------ ----- ------------------------------ --
AAAXoNAAGAAAJzeAAA 11201 Brooklyn                       NY
AAAXoNAAGAAAJzeAAB 75201 Dallas                         TX
AAAXoNAAGAAAJzeAAC 80401 Golden                         CO
AAAXoNAAGAAAJzeAAD 92101 San Diego                      CA
```

Foreign Keys

Put the relational in relational databases

What Is a Foreign Key



For a record to exist in the **COURSES** table, it must contain a valid **DEPARTMENT_CODE** value from the **DEPARTMENTS** table

Defining a Foreign Key

```
CREATE TABLE courses
(
    department_code     VARCHAR2(2)     NOT NULL,
    course_number       NUMBER(3,0)     NOT NULL,
    course_title        VARCHAR2(64)    NOT NULL,
    course_description  VARCHAR2(512)   NOT NULL,
    credits             NUMBER(3,1)     NOT NULL,
    CONSTRAINT pk_courses
        PRIMARY KEY (department_code, course_number),
    CONSTRAINT fk_courses_department_code
        FOREIGN KEY (department_code)
        REFERENCES departments (department_code),
);
```

Add a Foreign Key to an Existing Table

```
ALTER TABLE courses
    CONSTRAINT fk_courses_department_code
    ADD FOREIGN KEY (department_code)
    REFERENCES departments (department_code);
```

On Delete Cascade

```
CREATE TABLE courses
(
    department_code     VARCHAR2(2)     NOT NULL,
    course_number       NUMBER(3,0)     NOT NULL,
    course_title        VARCHAR2(64)    NOT NULL,
    course_description  VARCHAR2(512)   NOT NULL,
    credits             NUMBER(3,1)     NOT NULL,
    CONSTRAINT pk_courses
        PRIMARY KEY (department_code, course_number),
    CONSTRAINT fk_courses_department_code
        FOREIGN KEY (department_code)
        REFERENCES departments (department_code)
        ON DELETE CASCADE
);
```

On Delete Cascade

departments

department_code	name
MA	Math
PH	Physics
CS	Computer Science

courses

department_code	course_number	Title
MA	101	Calculus 1
MA	102	Calculus 2
PH	101	Physics 1
PH	102	Physics 2
CS	101	Intro to Programming

```
DELETE FROM departments
    WHERE department_code = 'PH';
```

On Delete Cascade

- **Without on delete cascade**

```
DELETE FROM courses WHERE department_code = ‘PH’;
DELETE FROM departments WHERE department_code = ‘PH’;
COMMIT;
```

- **To delete a primary key value, no child records can exist**

Deferred Constraints

```
CREATE TABLE courses
(
    department_code     VARCHAR2(2)     NOT NULL,
    course_number       NUMBER(3,0)     NOT NULL,
    course_title        VARCHAR2(64)    NOT NULL,
    course_description  VARCHAR2(512)   NOT NULL,
    credits             NUMBER(3,1)     NOT NULL,
    CONSTRAINT pk_courses
        PRIMARY KEY (department_code, course_number),
    CONSTRAINT fk_courses_department_code
        FOREIGN KEY (department_code)
        REFERENCES departments (department_code)
        DEFERRABLE
            INITIALLY IMMEDIATE
            -- INITIALLY DEFERRABLE
);
```

Using a Deferred Constraint

departments

department_code	name
MA	Math
PH	Physics
CS	Computer Science

```
set constraint
    fk_courses_depatment_code deferred;

UPDATE departments
    SET department_code = 'CO'
    WHERE department_code = 'CS';

UPDATE courses
    SET department_code = 'CO'
    WHERE department_code = 'CS';
COMMIT;
```

courses

department_code	course_number	Title
MA	101	Calculus 1
MA	102	Calculus 2
PH	101	Physics 1
PH	102	Physics 2
CS	101	Intro to Programming

Foreign Key Options Summary

- **Why are you having to cascade deletes or defer constraints?**
  - One time cleanup or typical business operation
- **Using on delete cascade**
  - Is deleting data really the correct decision?
  - Can the parent record be marked ‘inactive’
- **Using deferred constraints**
  - Consider a surrogate key for a primary key

Check Constraints

Check that a value is within a given **range**

Compare two values in **different columns**

Validate the format of a value using a **regular expression**

Check If Value is Within a Range

```
CREATE TABLE courses
(
    department_code     VARCHAR2(2)     NOT NULL,
    course_number       NUMBER(3,0)     NOT NULL,
    course_title        VARCHAR2(64)    NOT NULL,
    course_description  VARCHAR2(512)   NOT NULL,
    credits             NUMBER(3,1)     NOT NULL,
    CONSTRAINT pk_courses
        PRIMARY KEY (department_code, course_number),
    CONSTRAINT fk_courses_department_code
        FOREIGN KEY (department_code)
        REFERENCES departments (department_code),
    CONSTRAINT ck_courses_course_number
        CHECK (course_number BETWEEN 100 AND 999)
);
```

Check If Value is Within a Domain

```
CREATE TABLE states
(
    state_code          VARCHAR2(2)     NOT NULL,
    state_name          VARCHAR2(30)    NOT NULL,
    region              VARCHAR2(2)     NOT NULL,
    CONSTRAINT pk_states
        PRIMARY KEY (state_code),
    CONSTRAINT ck_state_regions
        CHECK (region IN ('NE', 'SE', 'MW', 'SC', 'NW', 'SW'))
);
```

Emulate a Boolean Column

```
CREATE TABLE students
(
    student_id        NUMBER(6)       NOT NULL,
    first_name        VARCHAR2(30)    NOT NULL,
    last_name         VARCHAR2(30)    NOT NULL,
    email_address     VARCHAR2(128)   NOT NULL,
    likes_ice_cream   NUMBER(1)       NULL,
    CONSTRAINT pk_students PRIMARY KEY (student_id),
    CONSTRAINT ck_studens_ice_cream
        CHECK (likes_ice_cream IN (0,1))
);
```

Compare Two Values

```
CREATE TABLE Orders
(
    order_id      NUMBER(9)     NOT NULL,
    customer_id   NUMBER(6)     NOT NULL,
    order_date    DATE          NOT NULL,
    ship_date     DATE          NULL,
    order_status  VARCHAR2(1)   NOT NULL,
    CONSTRAINT pk_orders
        PRIMARY KEY (order_id),
    CONSTRAINT ck_orders_order_ship_date
        CHECK (ship_date > order_date)
);
```

Validate Format Using a Regular Expression

```
CREATE TABLE zip_codes
(
    zip_code      VARCHAR2(5)       NOT NULL,
    city          VARCHAR2(30)      NOT NULL,
    state_code    VARCHAR2(30)      NOT NULL,
    CONSTRAINT pk_codes
        PRIMARY KEY (state_code),
    CONSTRAINT ck_zip_code_format
        CHECK (REGEXP_LIKE (zip_code, '^[0-9]{5}$') )
);
```

Add a Constraint to an Existing Table

```
ALTER TABLE zip_codes
    ADD CONSTRAINT ck_zip_codes_format
    CHECK (REGEXP_LIKE (zip_code, '^[0-9]{5}$') );
```

Disabling and Enabling Constraints

Disabling a constraint

```
ALTER TABLE courses
    DISABLE CONSTRAINT fk_courses_department_code;
```

Enabling a constraint

```
ALTER TABLE courses
    ENABLE CONSTRAINT fk_courses_department_code;
```

Constraints and Data Integrity

Multiple options are available for enforcing **data integrity**

Data integrity is critical to your **business data**

Why Use Database Constraints

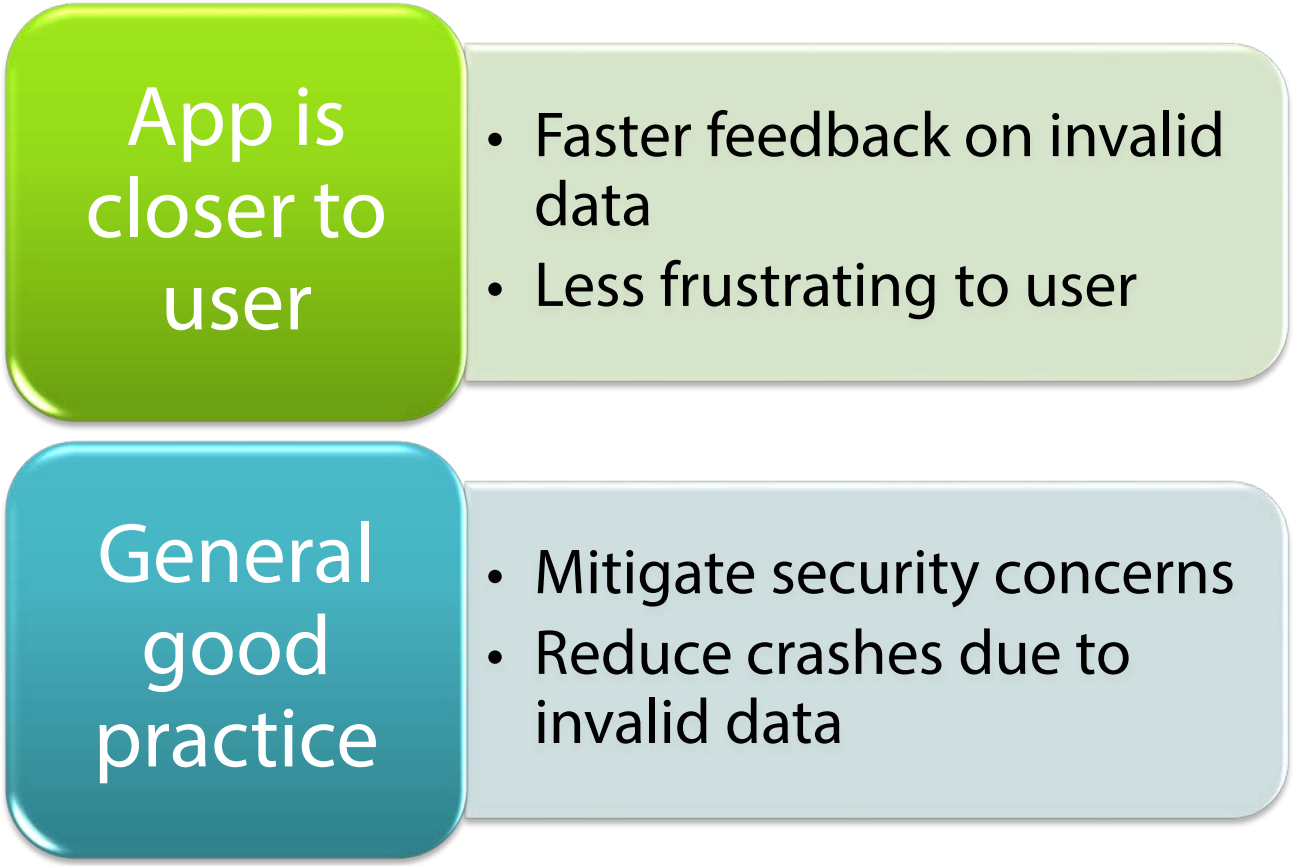
Clean Data

- Required for many business functions
- Can reduce the amount of coding needed to deal with "bad data"

Consistent Enforcement of Constraints

- All applications must comply with database constraints

Enforcing Constraints in Your Application



Constraints Strategy

Database Constraints

- One set of tools in an overall strategy
- Make use of all tools available

Validate both in Database and Applications

- Applications: provide immediate user feedback
- Database: consistently enforce rules across all applications

Performance Impacts

- Constraints can be checked in milliseconds
- Bad data take s along time to clean up

Summary

Table Storage Options

David Berry
http://buildingbettersoftware.blogspot.com/

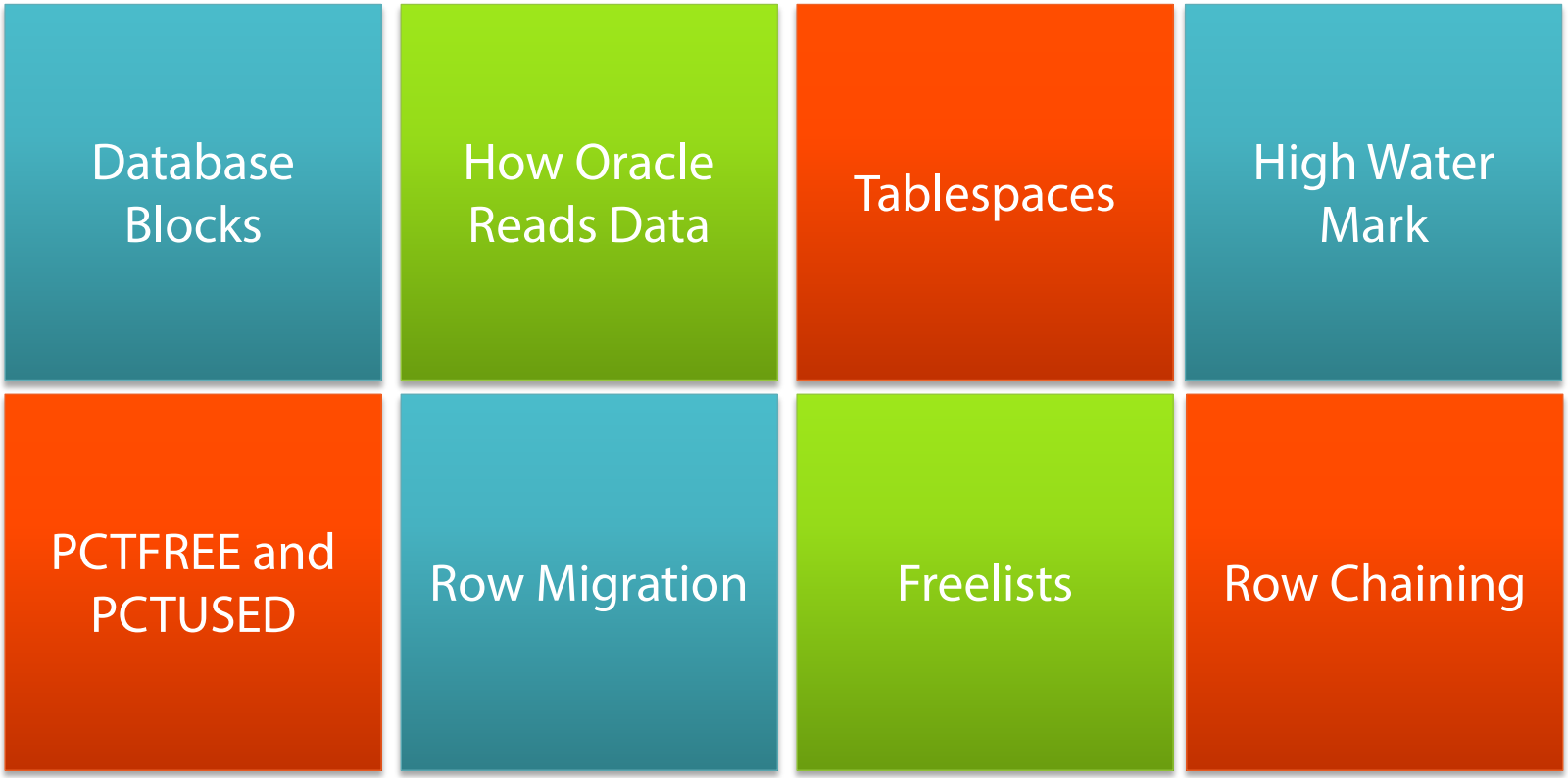
pluralsight
hardcore dev and IT training

Introduction

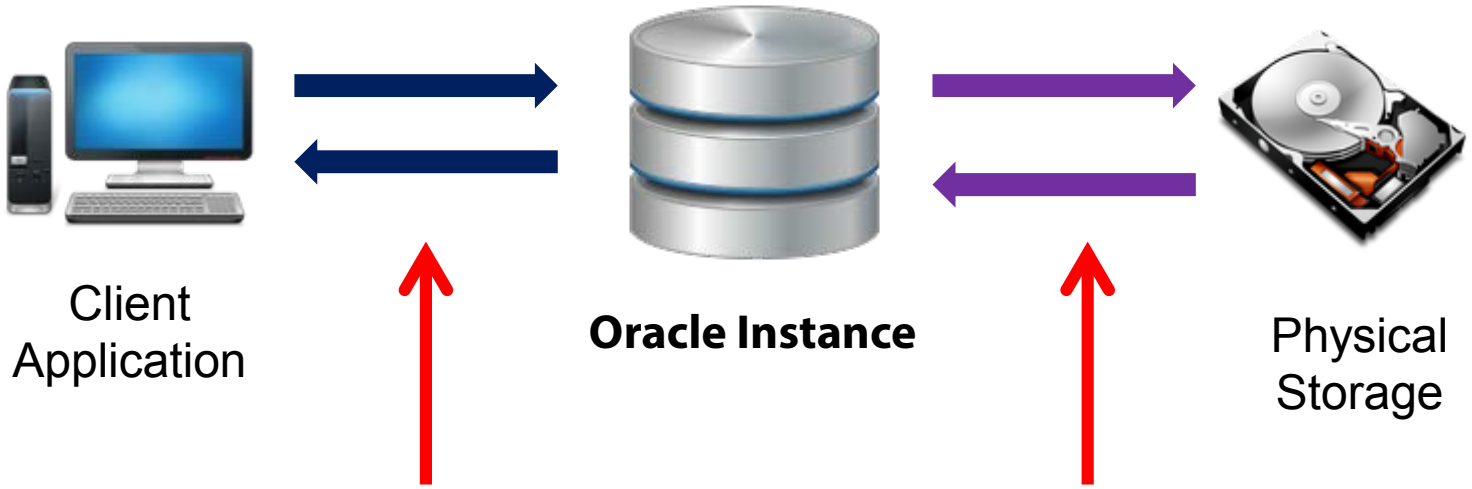
students table

student_id	first_name	last_name	phone
208718	Adam	Brady	360-592-7886
208723	George	Ward	216-952-0212
208728	Marcia	Garcia	630-239-4325
208741	Walter	Stumpf	419-500-5489
208749	Brenda	Brown	979-859-7105

Module Outline



Database Blocks



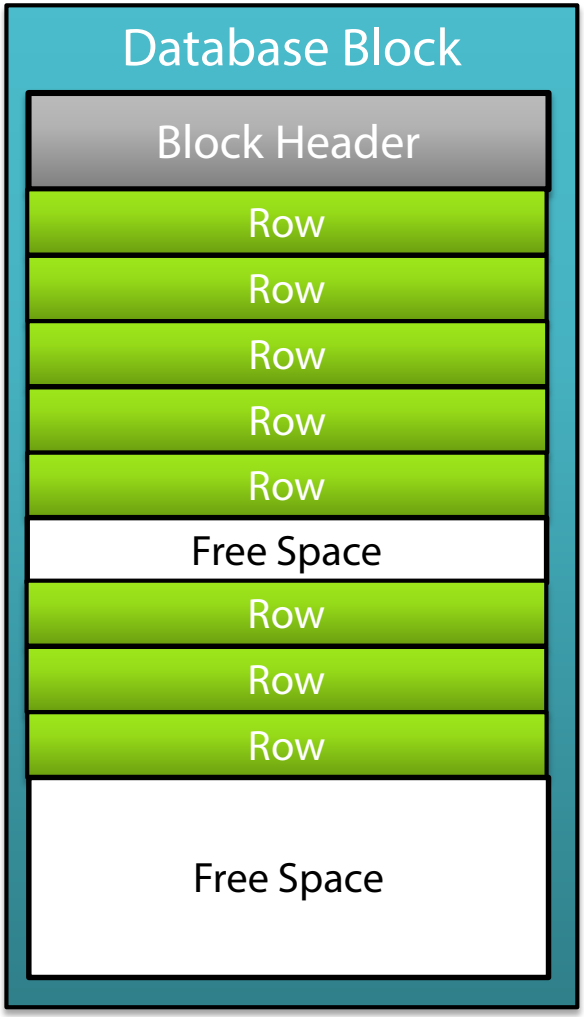
A data block is like a shipping container

What Is In a Block

Block Header
- Block type information
- Object assignment
- Row directory

Data
- Actual data for your table

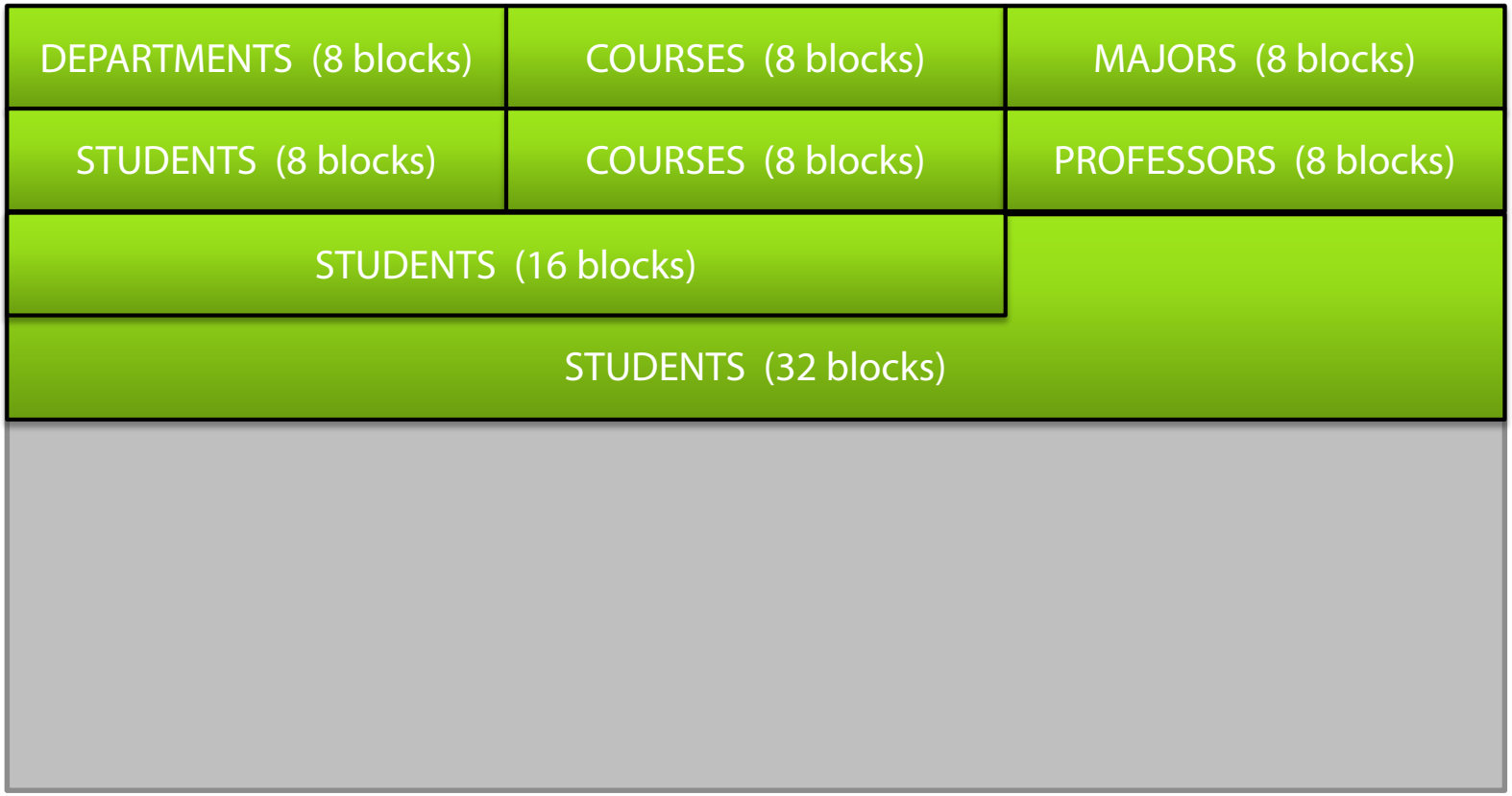
Free Space
- Allows for updates to rows



Oracle Storage Terminology

Data Block	Smallest level of storage in Oracle Can be between 2-32KB (8KB common)
Extent	Contiguous allocation of blocks for storing a specific type of data
Segment	Set of extents allocated for a specific database structure

Storage Allocation



How Oracle Performs Data Access

Full table scan

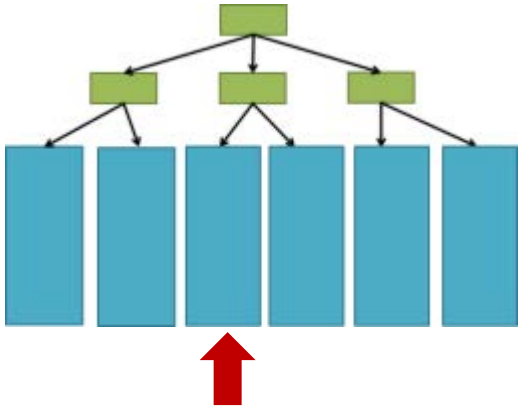
Index operation with row lookup

Full Table Scan

- Oracle scans through every block in the table
- Comparable to a linear search of an array

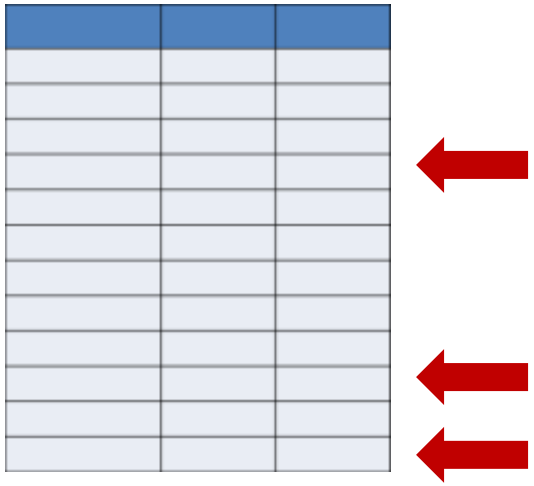
Index Operation and Row Lookup

Index operation



Oracle traverses the index tree to find matching keys

Row lookup in table



Oracle looks up the rows in the table by ROWID from index

Why Does This Matter?

Some situations **affect one data access** path, but **not the other**

Table Data and Sort Order

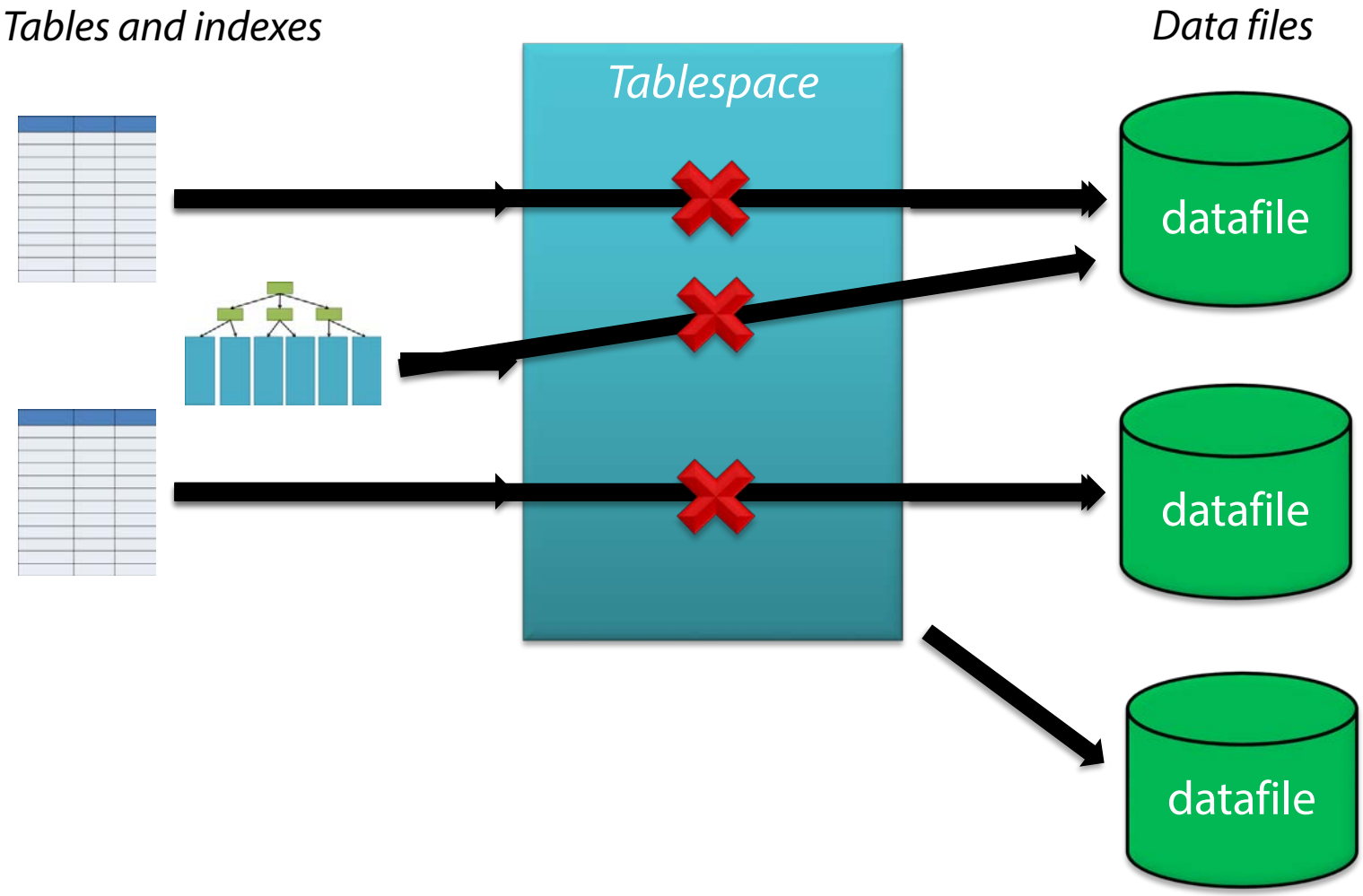
Data in a table has no implied order

- Oracle will insert rows wherever they fit
- Physical order on disk != Order data was inserted in

Data returned from queries is not sorted

- Order of results is not order data was inserted
- Two different executions can return same results in different order

Tablespace Introduction



Tablespace Block Size

Between 2 KB and 32 KB
(8 KB is a typical size)

Useful for estimating number of rows per block

Need to know to avoid row chaining

Segment Space Management

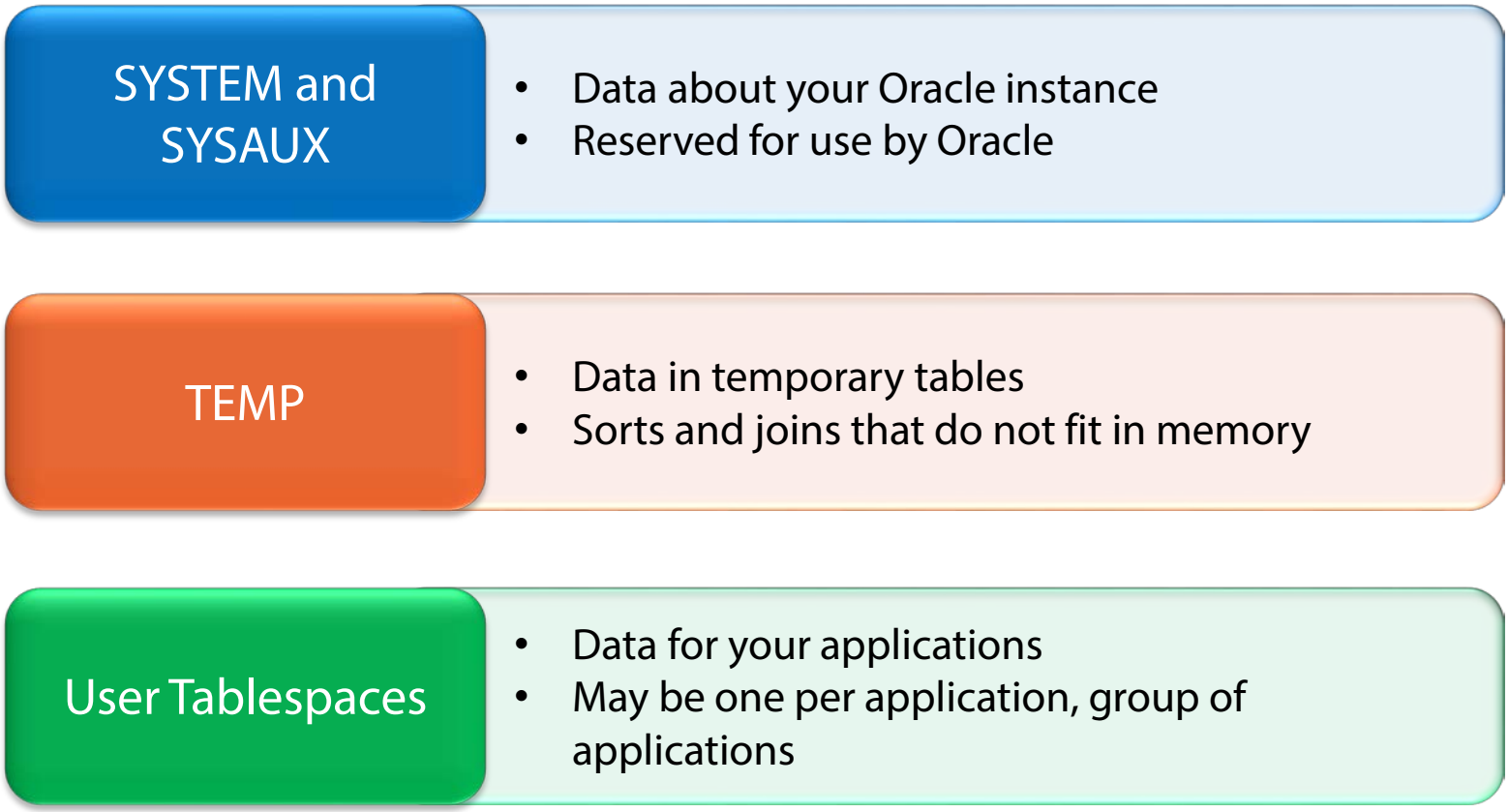
Manual Segment Space Management (MSSM)

- Original space management implementation
- You are responsible for managing all parameters for space management

Automatic Segment Space Management (ASSM)

- Introduced in Oracle 9i
- You specify value for PCTFREE
- Oracle manages all other parameters

Oracle Contains Multiple Tablespaces



Default Tablespace

Every user has a **default** tablespace

Some objects may be created in an **alternate** tablespace

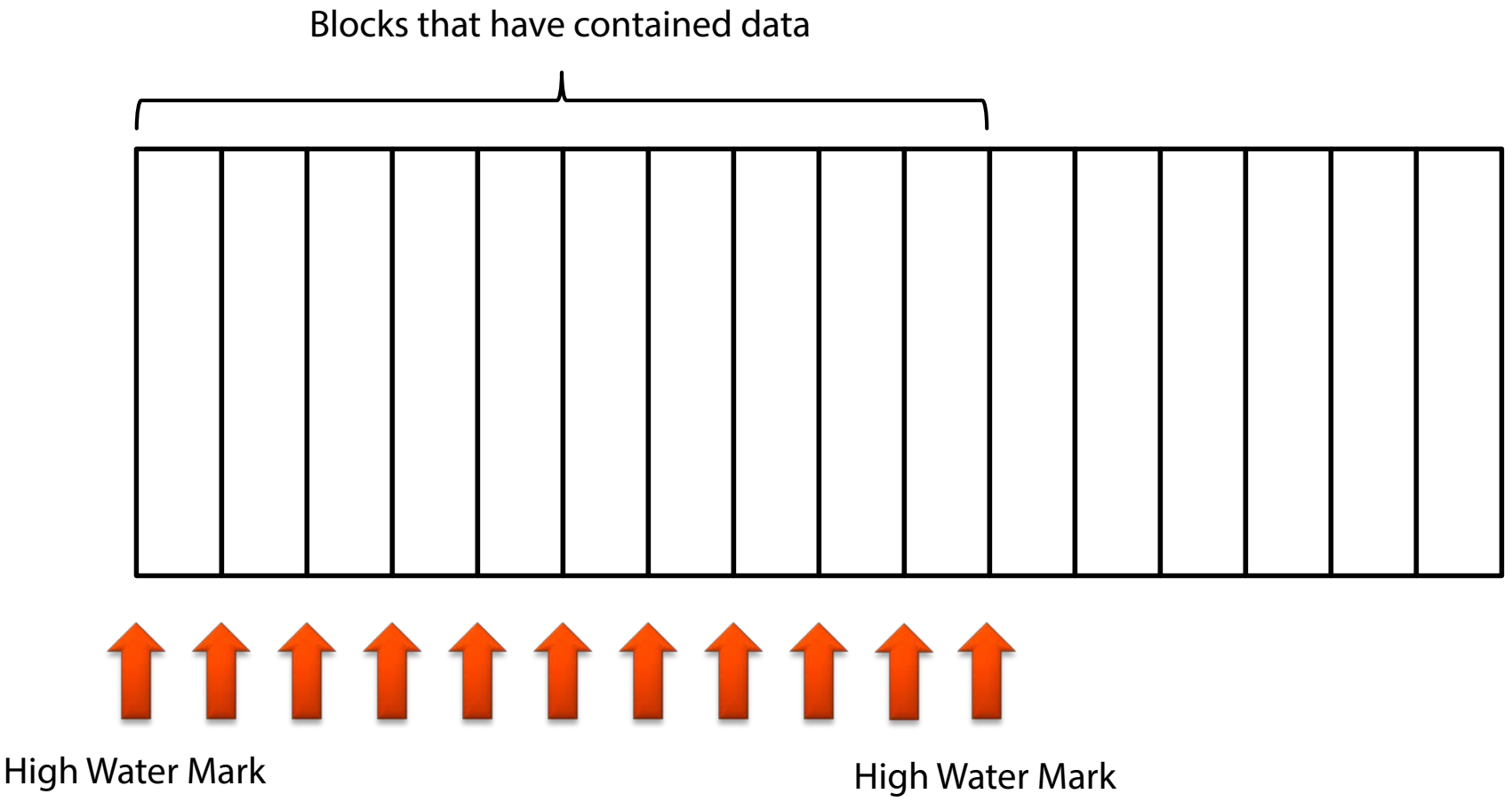
Specifying a Tablespace for a Table

```
CREATE TABLE course_enrollments
(
    course_enrollment_id     NUMBER     NOT NULL,
    course_offering_id       NUMBER(20)   NOT NULL,
    student_id               NUMBER(10)   NOT NULL,
    grade_code               VARCHAR2(1)  NULL,
    CONSTRAINT pk_course_enrollments
        PRIMARY KEY (course_enrollment_id),
    CONSTRAINT fk_enrollments_offerings
        FOREIGN KEY (course_offering_id)
        REFERENCES course_offerings (course_offering_id),
    CONSTRAINT fk_enrollments_student
        FOREIGN KEY (student_id) REFERENCES students (student_id),
    CONSTRAINT fk_enrollments_grades
        FOREIGN KEY (grade_code) REFERENCES grades (grade_code)
)
TABLESPACE student_data;
```

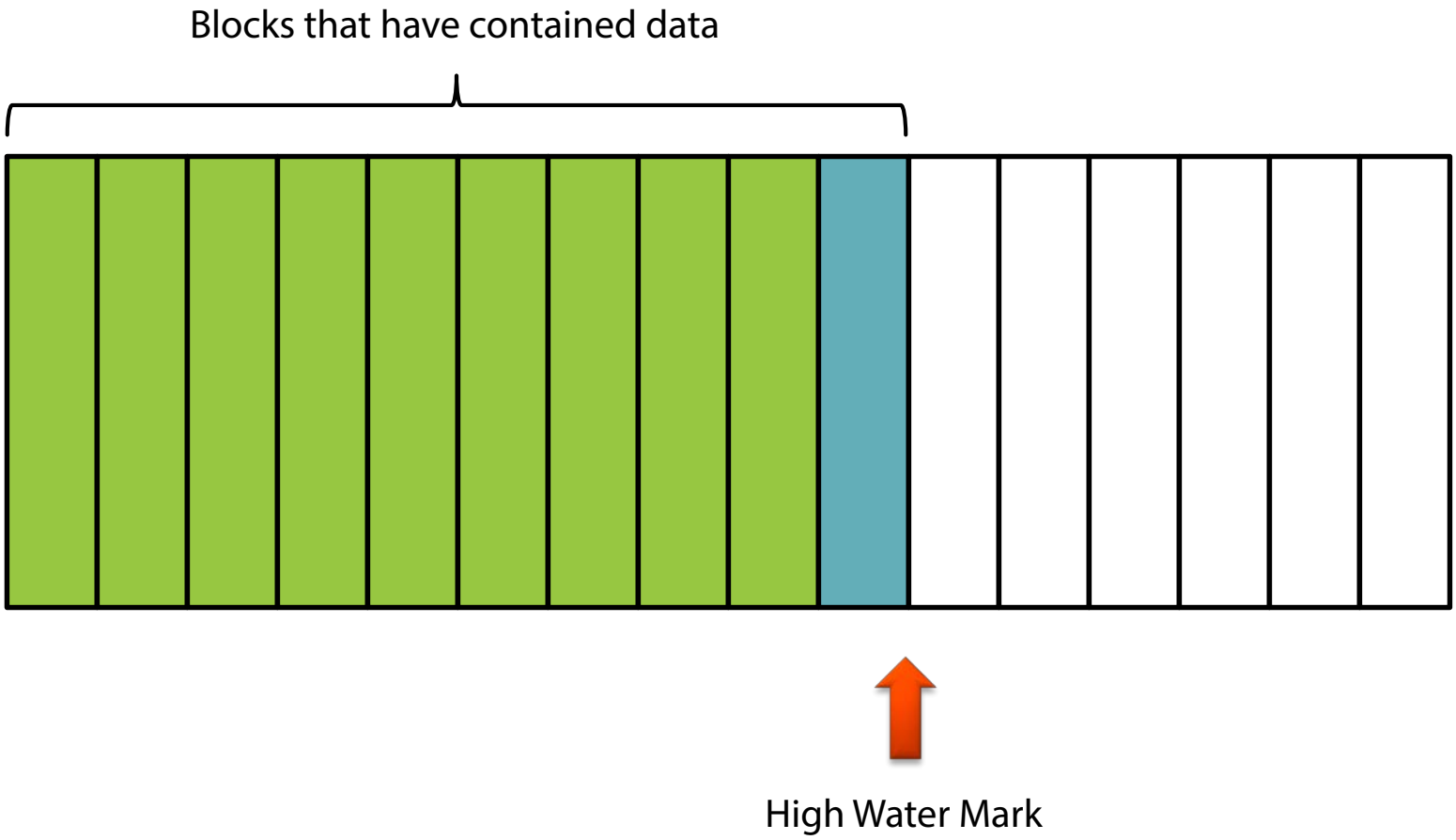
High Water Mark

Highest numbered block that has ever contained data in a table

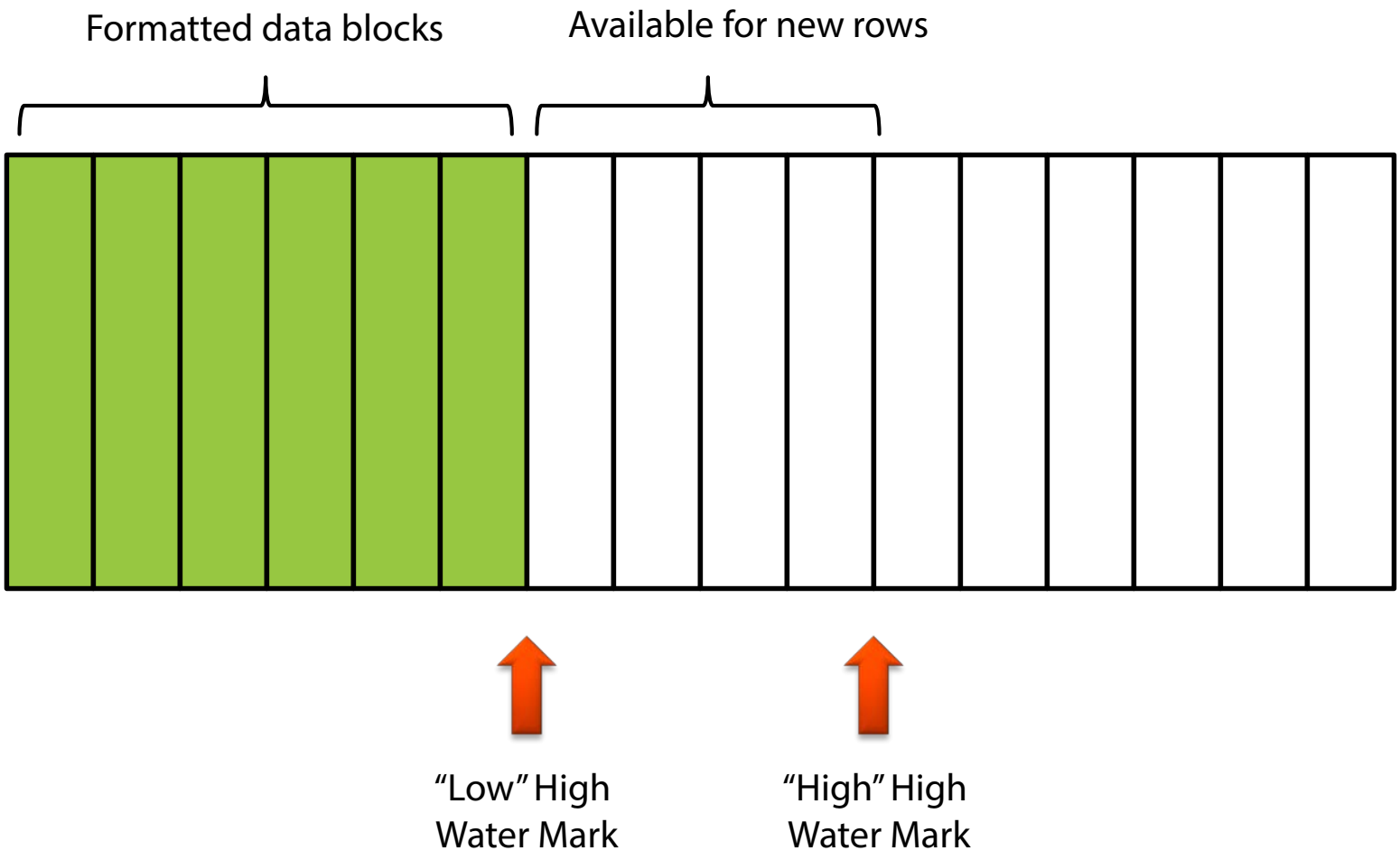
High Water Mark



High Water Mark – Deleting Rows



High Water Mark – ASSM

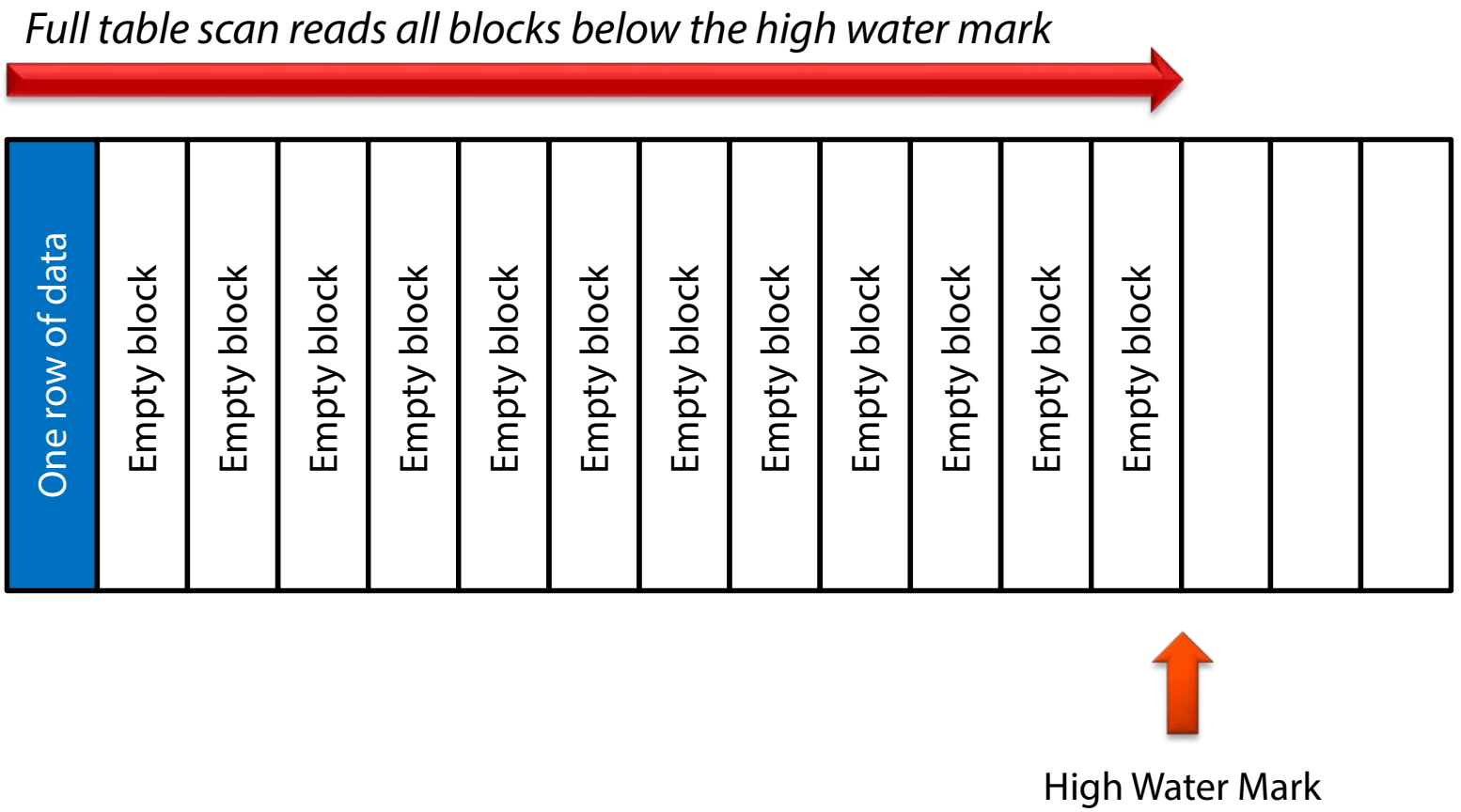


High Water Mark Impacts

During a **full table scan**, Oracle reads all blocks below the **high water mark**

If many of these blocks are **empty**, this is a lot of **wasted work**

High Water Mark and Full Table Scans



High Water Mark Impacts

Should I Worry?

- Generally, no
- Natural to have a few empty blocks below the high water mark

One scenario to monitor

- Tables that have a large number of rows inserted and deleted
- Subsequent processing uses many fewer records
- Consider TRUNCATE instead of DELETE

A data block is “full”

What does this really mean?

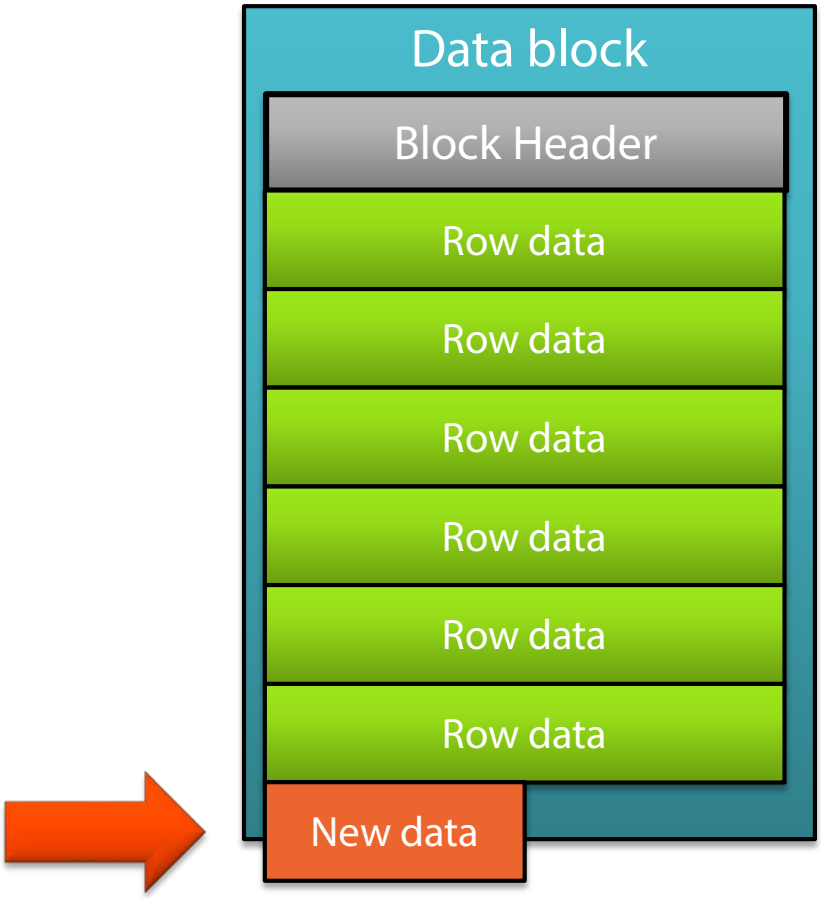
Storage Implications of Updating Rows

student_id	first_name	last_name	phone	email_address	major_id
208718	Adam	Brady	360-592-7886	AdamBBrady@dayrep.com	3
208723	George	Ward	216-952-0212	GKWard@superrito.com	5
188728	Marcia	Garcia	630-239-4325	MarciaDGarcia@rhyta.com	8
208741	Walter	Stumpf	NULL	WalterCStumpf@einrot.com	12

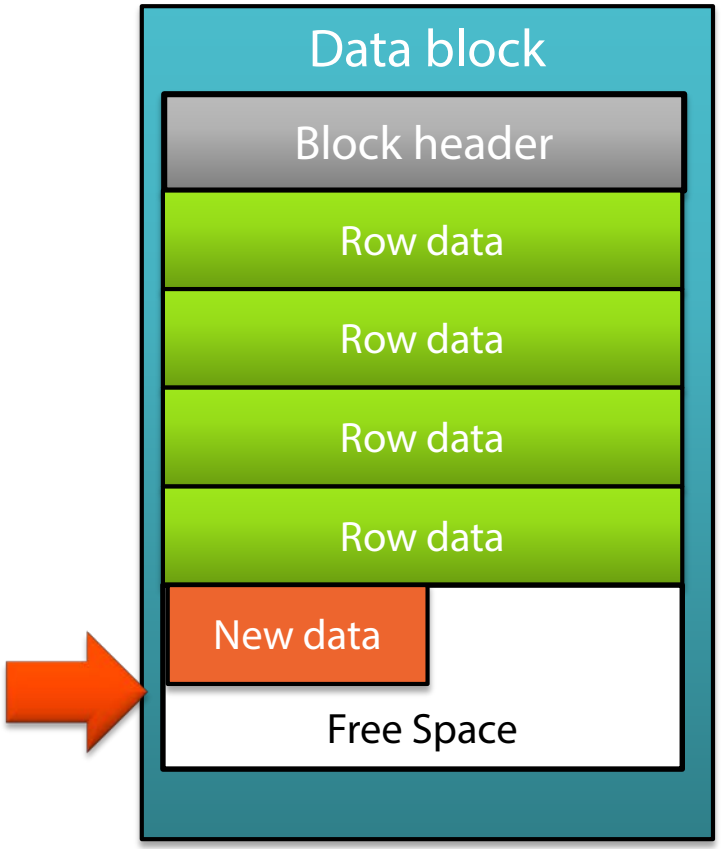
```
UPDATE students
    SET last_name = 'Anderson'
    WHERE student_id = 188728;

UPDATE students
    SET phone = '419-500-5489'
    WHERE student_id = 208741;
```

PCTFREE Introduction

No space is left for row to expand when the row is updated



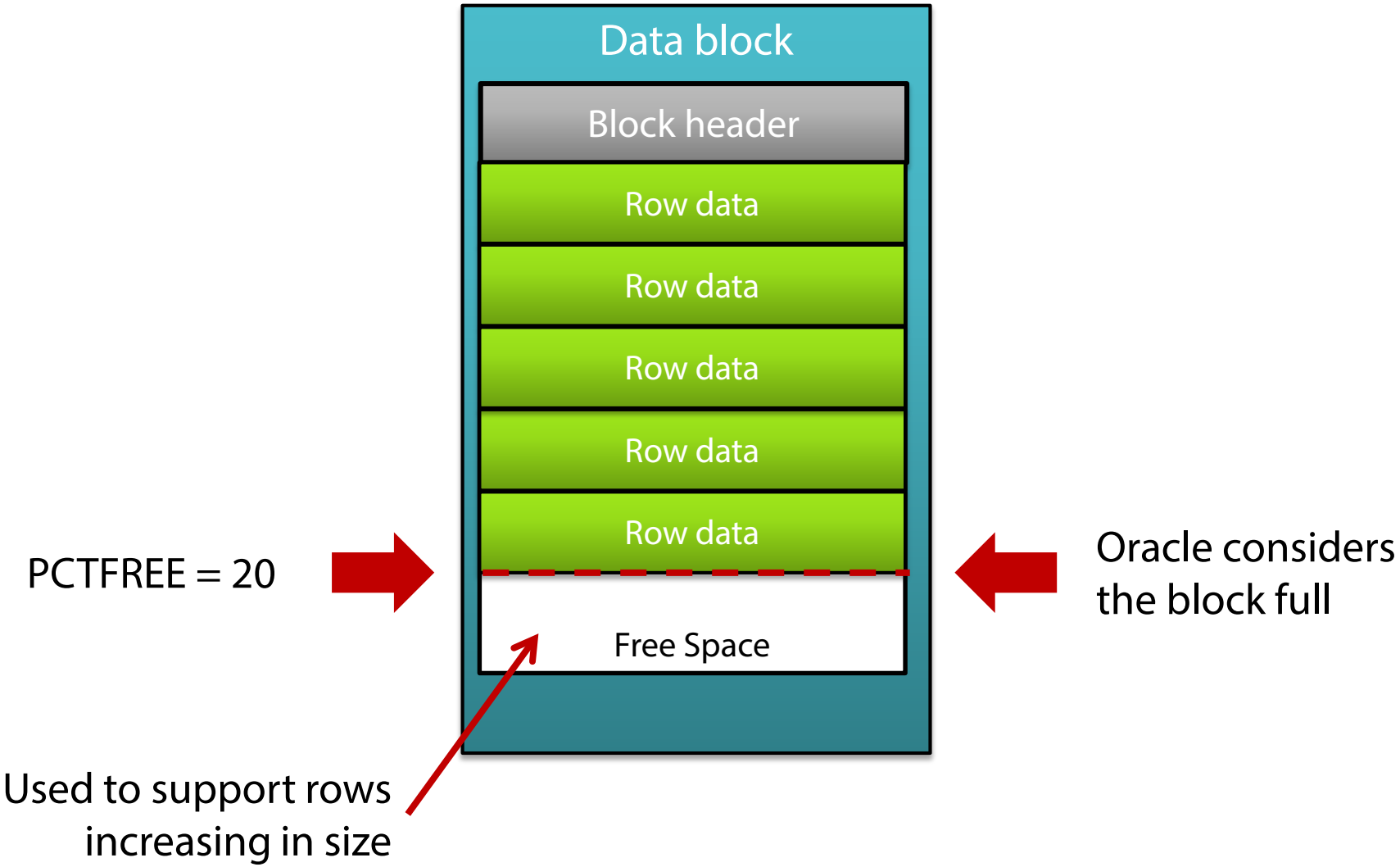
PCTFREE Introduction



Space is available for row to expand on update

Amount of space left is controlled by PCTFREE parameter

PCTFREE Example



Setting PCTFREE

- Default value is 10 percent

- Specify using PCTFREE in storage clause

```
CREATE TABLE students
(
    student_id              NUMBER(10)     NOT NULL,
    first_name              VARCHAR2(40)   NOT NULL,
    middle_name             VARCHAR2(40)   NOT NULL,
    last_name               VARCHAR2(40)   NOT NULL,
    gender                  VARCHAR2(1)    NOT NULL,
    street_address          VARCHAR2(50)   NOT NULL,
    city                    VARCHAR2(30)   NOT NULL,
    state                   VARCHAR2(2)    NOT NULL,
    email                   VARCHAR2(80)   NOT NULL,
    …
)
PCTFREE 20;
```

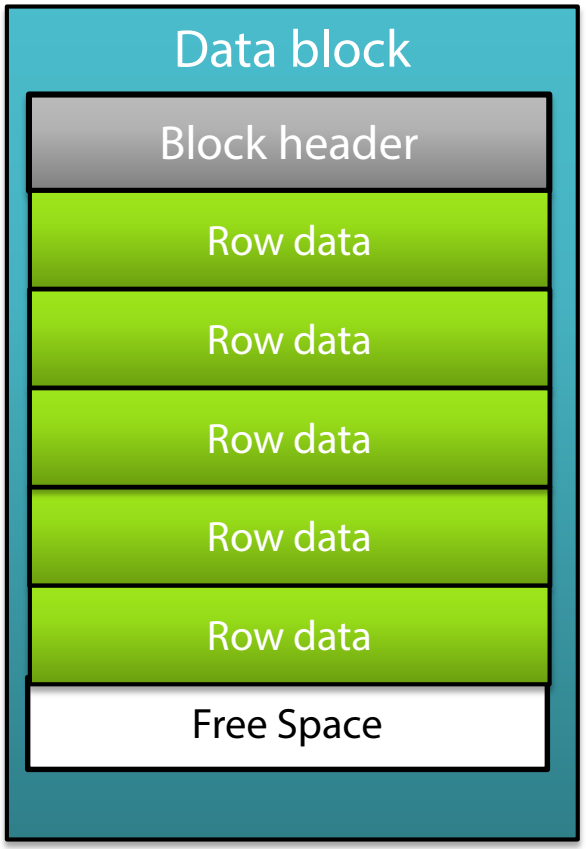
PCTFREE

- Applies to both ASSM an MSSM tablespaces
- Need to understand table data, update patterns to set PCTFREE
- PCTFREE too low – row migrations will occur on updates
- PCTFREE too high – space will be wasted in each block

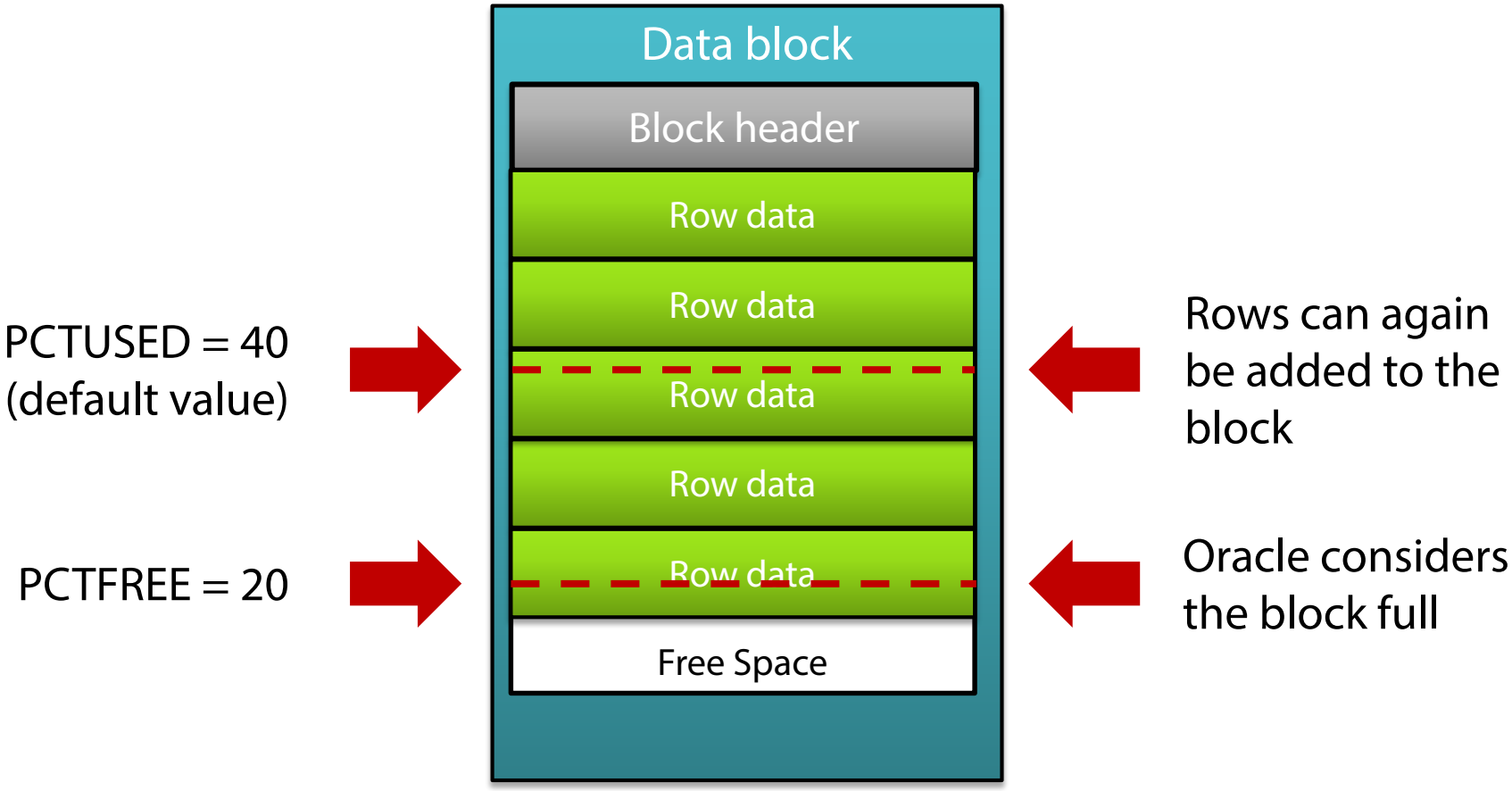
PCTUSED

Only applies to MSSM tablespaces

Threshold at which Oracle will once again allow inserts into the block



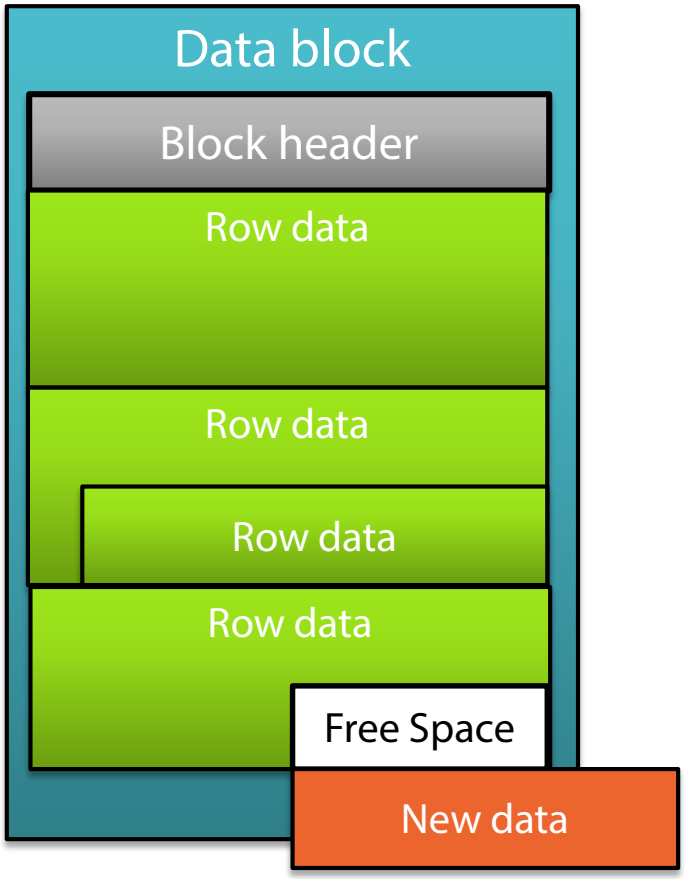
PCTUSED Example



Setting PCTUSED

```
CREATE TABLE students
(
    student_id              NUMBER(10)     NOT NULL,
    first_name              VARCHAR2(40)   NOT NULL,
    middle_name             VARCHAR2(40)   NOT NULL,
    last_name               VARCHAR2(40)   NOT NULL,
    gender                  VARCHAR2(1)    NOT NULL,
    street_address          VARCHAR2(50)   NOT NULL,
    city                    VARCHAR2(30)   NOT NULL,
    state                   VARCHAR2(2)    NOT NULL,
    email                   VARCHAR2(80)   NOT NULL,
    …
)
PCTUSED 50;
```

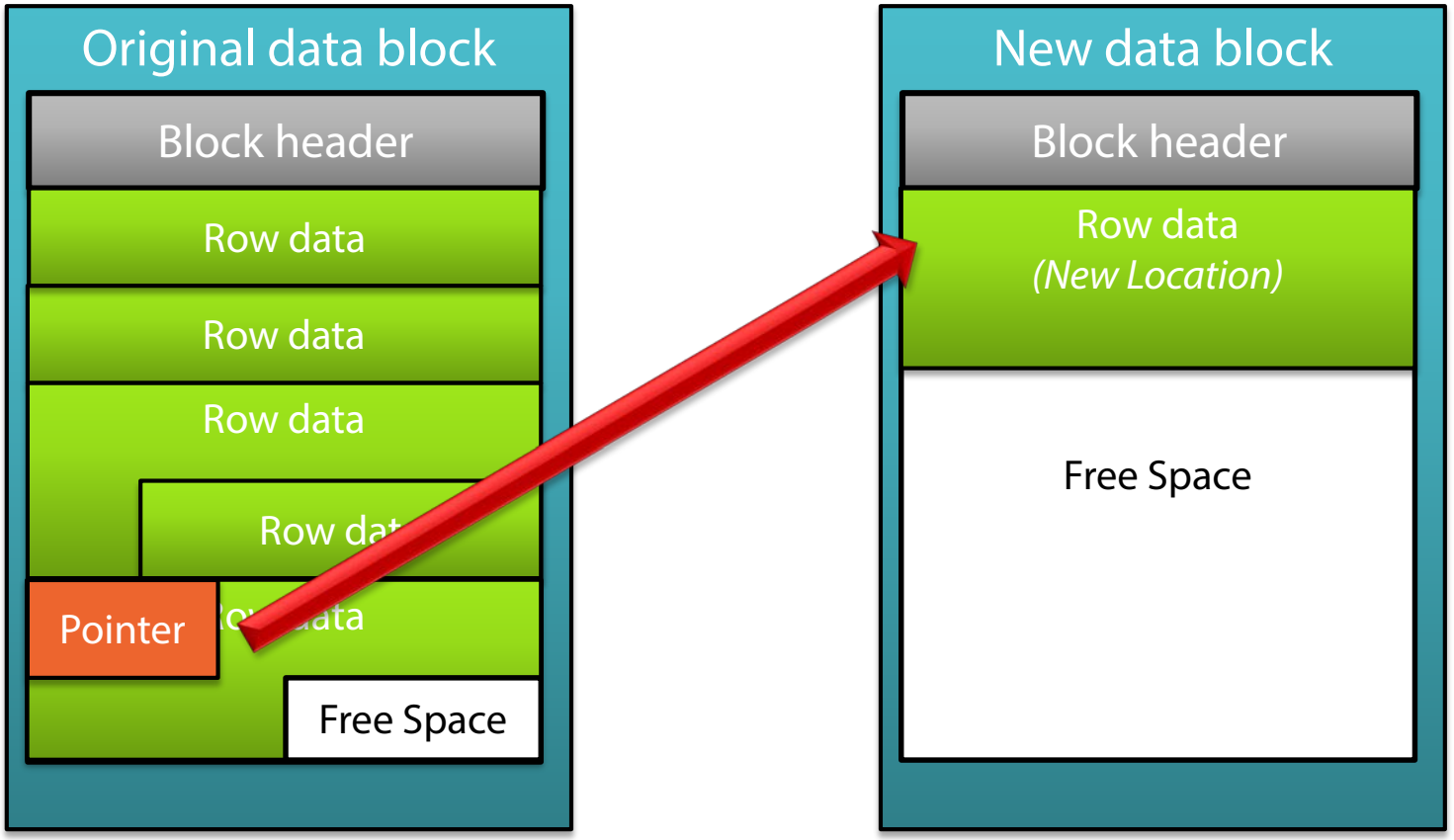
When Does Row Migration Occur?



Updated data will not fit in available free space of block

Oracle will **migrate** the **entire row** to a **different** data block

Row Migration



Pointer points from old location to new location

Impacts of Row Migration

- Row lookup operations will now require two IO's
- Full table scans do not incur a penalty
- Row can be migrated multiple times but Oracle updates the original pointer

Types of Row Migration

PCTFREE Too Low

- Insufficient space for rows to expand into
- Many rows will be migrated through normal updates
- Something to be avoided

Natural Row Migration

- Natural for a small percentage of rows to migrate
- Nothing to worry about

Detecting Row Migration

Extra work is coming from row migrations?

```
SELECT name, value
  FROM v$sysstat
 WHERE name IN ('table fetch continued row',
     'table fetch by rowid');
```

Migrated and chained rows per table

```
SELECT table_name, chain_cnt, num_rows, last_analyzed
     FROM user_tables;
```

Practical Guidance on PCTFREE

PCTFREE too low

Row migrations will occur

PCTFREE too high

Waste space with partially filled blocks

Using a Low PCTFREE Value

Table Characteristics

- No update activity (inserts only)
- Update will not affect the size of a row

Examples

- Logging tables
- Some data warehouse applications

Using a High PCTFREE Value

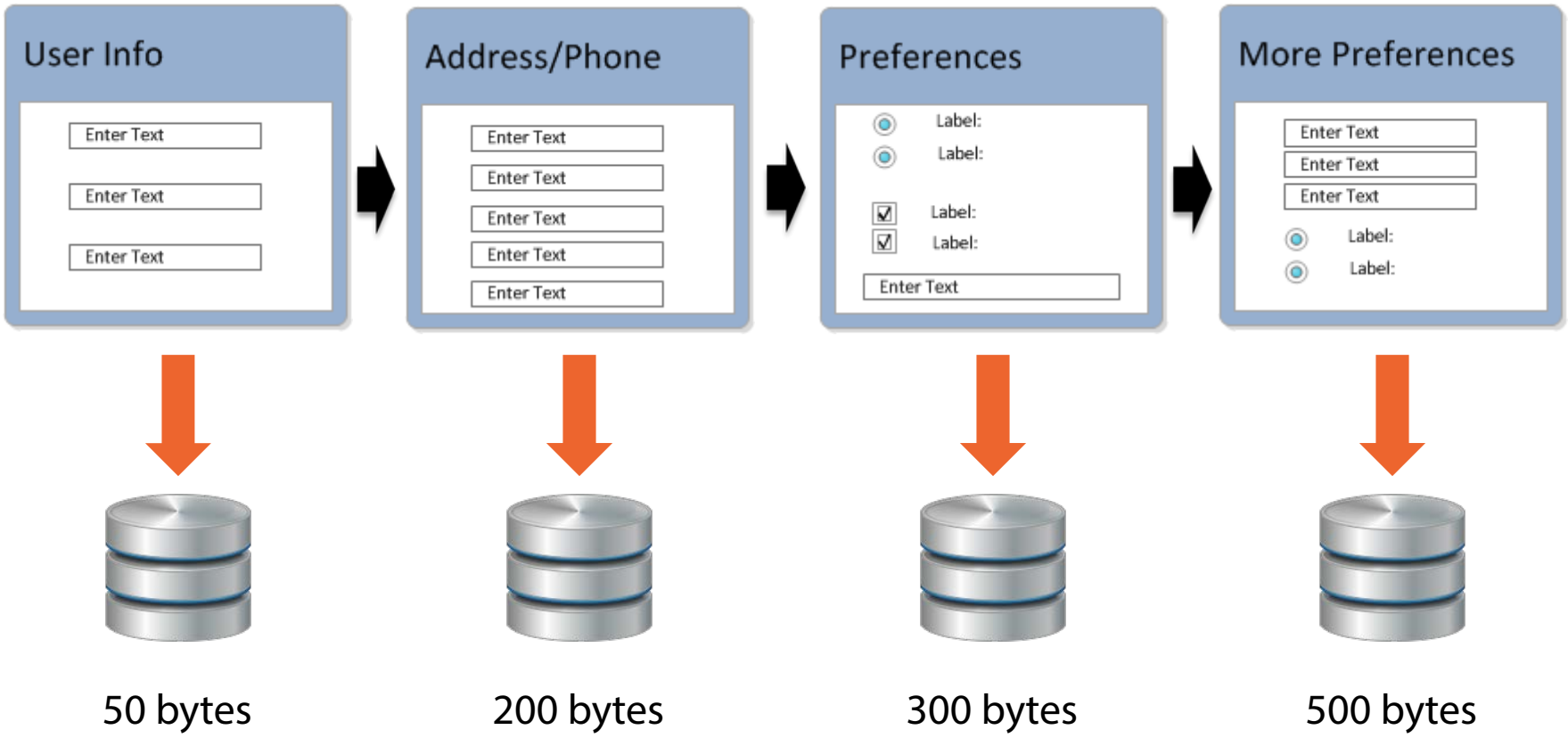
Table Characteristics

- Data added to rows incrementally
- Updates increase row size

Examples

- Tables seeded with initial value, other data added over time

A Sample Scenario



PCTFREE Considerations

Know how your application inserts and updates data

Don’t be afraid to run a test

Freelists

Definition

List of blocks that have free space that Oracle can use to insert new rows into

ASSM

Freelists are automatically managed

Major advantage of ASSM

MSSM

Default number of freelists is one

Can be a bottleneck for concurrent DML

Setting FREELISTS at Table Creation

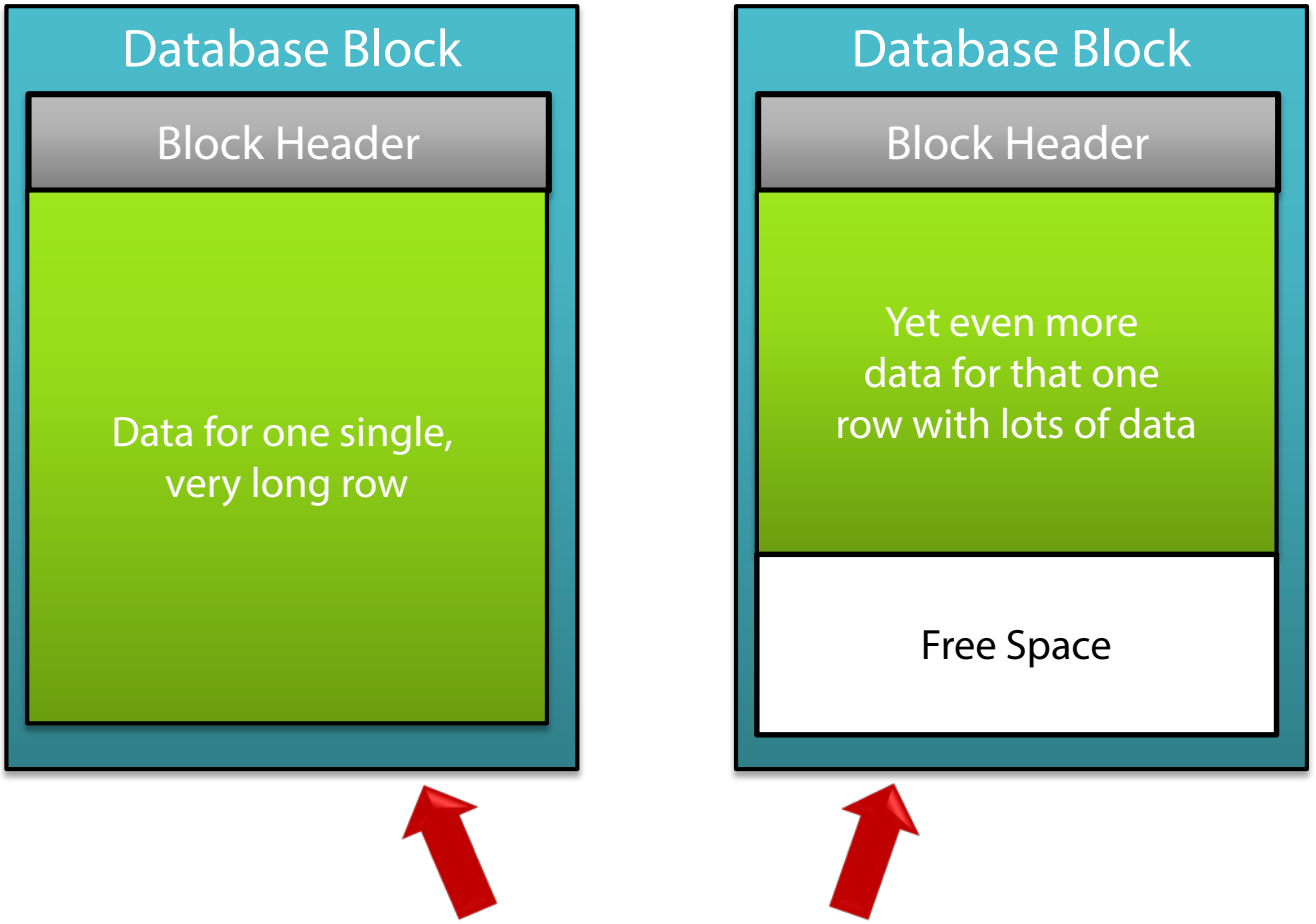
```
CREATE TABLE course_enrollments
(
    course_enrollment_id     NUMBER        NOT NULL,
    course_offering_id       NUMBER(20)    NOT NULL,
    student_id               NUMBER(10)    NOT NULL,
    grade_code               VARCHAR2(1)   NULL,
    …
)
STORAGE (FREELISTS 12);
```

Changing Freelists on an Existing Table

```
ALTER TABLE course_enrollments
    STORAGE (FREELISTS 12);
```

- Correct table where freelists was set incorrectly
- Perform a bulk load in parallel

Row Chaining



Two IO operations will be required to read this row from disk

Table With Row Chaining

Tablesapce block size

Value between 2 – 32 KB

Assume for example 4 KB

Sample Table Definition

```
CREATE TABLE products
(
    product_id      NUMBER(8)       NOT NULL,
    sku             VARCHAR2(32)    NOT NULL,
    name            VARCHAR2(128)   NOT NULL,
    description     VARCHAR2(4000)  NOT NULL,
    tech_specs      VARCHAR2(4000)  NOT NULL
);
```

Detecting Chained Rows

Tables at risk for row chaining

- Tables with lots of columns (>100 columns)
- Tables with large sized CHAR/VARCHAR2 columns

Statistics on chained and migrated rows

```
-- Gather statistics on the table
EXEC DBMS_STATS.GATHER_TABLE_STATS(
    ownname => '<<Table Owner Name>>',
    tabname => '<<Table Name>>');


-- Chain count is sum of chained and migrated rows
SELECT table_name, chain_cnt, num_rows, last_analyzed
     FROM user_tables;
```

Identifying Chained Rows

```
-- Script from $ORACLE_HOME/rdbms/admin/utlchain.sql
CREATE TABLE chained_rows
(
  owner_name         varchar2(128),
  table_name         varchar2(128),
  cluster_name       varchar2(128),
  partition_name     varchar2(128),
  subpartition_name  varchar2(128),
  head_rowid         rowid,
  analyze_timestamp  date
);

-- Analyze the table
ANALYZE TABLE <<table name>> LIST CHAINED ROWS;
```

Solutions To Row Chaining

Use a sufficient block size for your table data

- **Estimate** row sizes for each table from **logical data model**
- Specify the **tablespace block size** based on these estimates
- **Move** tables with large rows to a **new tablespace**

Vertically Partition Tables

product_id	sku	name	summary	product_description	technical_specs	product_review
400001	HD-100	256 GB Solid State Hard Drive	….	….	….	….
600483	GC-862	GraphicsStar! Graphics Card	….	….	….	….
7839390	MO-101	27” Flat Screen Monitor – 1080P	…..	….	….	….
8920201	MS-173	Wireless Mouse	…..	….	….	….

Vertically Partition Tables

product_id	sku	name	summary	product_description
400001	HD-100	256 GB Solid State Hard Drive	….	….
600483	GC-862	GraphicsStar! Graphics Card	….	….
7839390	MO-101	27" Flat Screen Monitor – 1080P	…..	….
8920201	MS-173	Wireless Mouse	…..	….

Foreign key between tables

product_id	technical_specs	product_review
400001	….	….
600483	….	….
7839390	….	….
8920201	….	….

Module Summary

- Data storage using data blocks
- How Oracle accesses data
- Tablespaces
- PCTFREE
- Row Migration
- Freelists
- Row Chaining

Oracle Database Concepts Guide

http://bit.ly/Ora12cDatabaseConcepts

Managing Tables

David Berry
http://buildingbettersoftware.blogspot.com/

Databases Require Changes Over Time

Adding columns

Changing data types

Renaming objects

Module Contents

Oracle Data Dictionary

Series of views containing metadata about all database objects in Oracle

Data Dictionary Course Content Focus

Graphical presentation varies by tool

Become familiar with feature set of tool you use

Be aware of underlying views

Use for writing custom queries, customizing output format

Three Sets of Views

dba_*	Information about all objects in the database
user_*	Information about objects owned by the current user
all_*	Information about all objects the current user has permission to

USER_OBJECTS View

Definition

Contains	An entry for every database object (table, view, index, stored procedure, etc…)
Oracle Documentation	http://bit.ly/Ora12cAllObjects

Key columns

object_name	Name of the table, view, index, etc…
object_type	Indicated if this object is a table, view, etc…
created	Timestamp of when this object was created
last_ddl_time	Timestamp of when the definition of this object was last changed
status	Indicated if the object is valid or invalid

USER_TABLES View

Definition

Contains	One row for each table
Oracle Documentation	http://bit.ly/Ora12cAllTables

Key columns

table_name	Name of the table
num_rows	Number of rows in the table
blocks	Number of blocks that have been used in the table
avg_row_len	Average length in bytes of a row in the table
last_analyzed	Date that stats were most recently gathered against the table

USER_TAB_COLS View

Definition

Contains	An entry for every column in a table or a view
Oracle Documentation	http://bit.ly/Ora12cAllTabCols

Key columns

table_name	Name of the table or view the column belongs to
column_name	Name of the column
data_type	Data type of this column
num_distinct	Number of distinct values in this column
num_nulls	Number of rows in this column with a null value

USER_CONSTRAINTS View

Definition

Contains	An entry for each constraint (primary key, foreign key, check constraint)
Oracle Documentation	http://bit.ly/Ora12cAllConstraints

Key columns

table_name	Name of the table the constraint is associated with
constraint_name	Name of the constraint
constraint_type	Code that indicates the type of the constraint
r_constriant_name	For foreign keys, the name of the unique constraint in the parent table
deferrable	Indicates if the constraint can be deferred

USER_INDEXES View

Definition

Contains	An entry for each index
Oracle Documentation	http://bit.ly/Ora12cAllindexes

Key columns

table_name	Name of the table the index is on
index_name	Name of index
uniqueness	Indicated if this is a unique index
distinct_keys	Number of distinct keys in the index
last_analyed	Date that stats were most recently gathered against the index

USER_IND_COLUMNS View

Definition

Contains	An entry for each column in an index
Oracle Documentation	http://bit.ly/Ora12cAllIndColumns

Key columns

table_name	Name of the table the index is on
index_name	Name of index
column_name	Name of the column
column_position	Position of the column in the index

Data Dictionary Summary

Invest some time in learning what is available

Most commonly used views

- http://bit.ly/Ora12cDataDictionaryViews

Life of a SQL Statement

Parse Phase

Check SQL Syntax
Check Object Permissions
Query Rewrite Process

Query Optimization

Evaluate Statistics
Create Execution Plan

Execution Phase

Read Blocks
Filter Rows
Sort Data

How Are Database Statistics Used?

By the Oracle Optimizer to determine the most efficient plan for your SQL statement

To estimate how much processing is required for each option

How Are Database Statistics Gathered?

Automatically

- Scheduled by Oracle
- Oracle detects objects with significant changes

Manually

- Executed by user
- Used after large insert/delete operation
- Useful in test environments

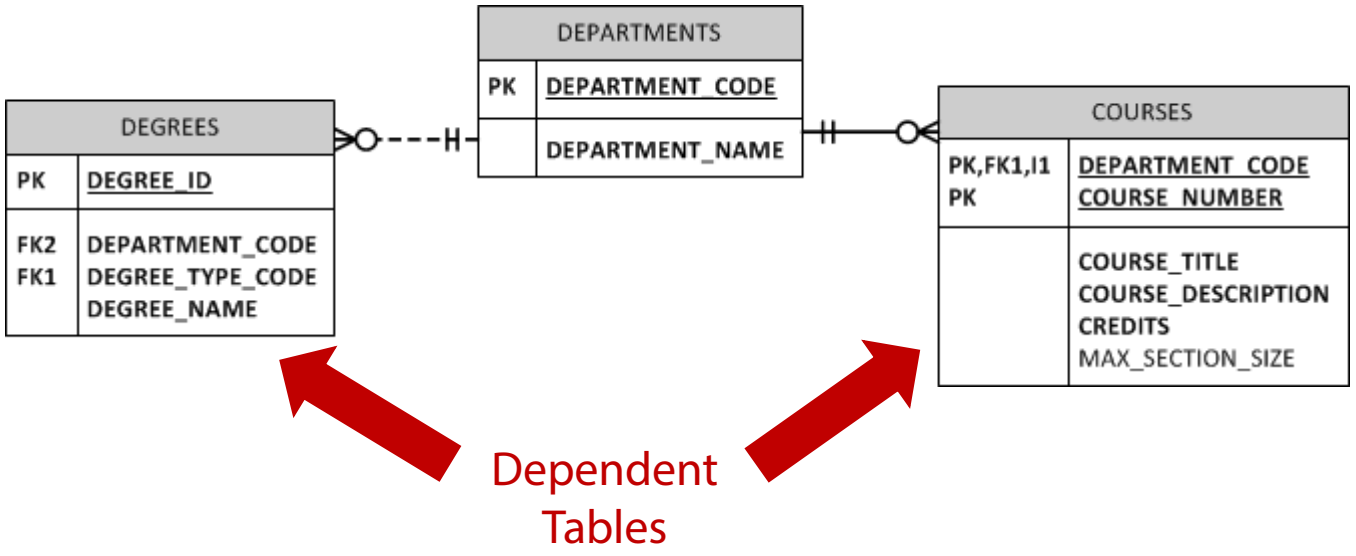
Gathering Statistics Manually

```
-- Update for one table and all indexes on table
EXEC DBMS_STATS.GATHER_TABLE_STATS(
    ownname => '<<Table Owner Name>>',
    tabname => '<<Table Name>>',
    estimate_percent => <<percent>>);


-- Update for all objects in a schema
EXEC DBMS_STATS. GATHER_SCHEMA_STATS(
    ownname => '<<Schema Owner Name>>',
    estimate_percent => <<percent>> );
```

http://bit.ly/Orac12cDbmsStats

Dropping a Table

```
-- Basic DROP TABLE syntax
DROP TABLE departments;
```



Foreign Keys and Dropping a Table

```
-- Drop dependent tables first
DROP TABLE degrees;
DROP TABLE courses;

-- Now drop the table of interest
DROP TABLE departments;
```

- Problematic if the dependent tables contain more dependencies
- We don't want to drop a significant portion of the database

Drop Table With Cascade Constraints

```
-- Drop table and foreign keys that reference this table
DROP TABLE departments CASCADE CONSTRAINTS;
```

- **Dependent tables stay intact**
  - Only their foreign keys are dropped

- **Possible to recreate the table, re-add foreign keys**

Renaming a Table

```
-- Syntax to rename an object in Oracle
-- RENAME old_name TO new_name;

RENAME departments TO depts;
```

- **Oracle will automatically transfer all constraints and indexes**
- **Views, stored procedures, functions, synonyms will become invalid**
  - Find invalid objects using all_objects view
  - You need to manually fix these

Renaming a Table

```
-- Syntax to rename an object in Oracle
-- RENAME old_name TO new_name;

RENAME departments TO depts;
```

- **Oracle will automatically transfer all constraints and indexes**
- **Views, stored procedures, functions, synonyms will become invalid**
  - Find invalid objects using all_objects view
  - You need to manually fix these

Making a Table Read Only

Place a table in read only mode

```
ALTER TABLE table_name READ ONLY;
```

- No DML allowed (INSERT, UPDATE, DELETE)
- TRUNCATE not allowed
- Cannot add, modify, remove columns

Return a table to read write mode

```
ALTER TABLE table_name READ WRITE;
```

Adding Columns to Tables

Frequent need over the lifetime of a database

Required capability to respond to **changing business needs**

Adding Columns to a Table

```
ALTER TABLE students ADD
(
  immunization_form_received   VARCHAR2(1) DEFAULT 'N' NOT NULL,
  immunization_form_date       DATE        NULL
);
```

Adding Columns With a Default Value

```
ALTER TABLE students ADD
(
  immunization_form_received   VARCHAR2(1) DEFAULT 'N' NOT NULL,
  immunization_form_date       DATE        NULL
);
```

- Oracle substitutes the default value at query time for existing rows
- Existing rows do not have to be updated

Adding Columns – Breaking Insert Statements

Insert statement without column names

```
INSERT INTO courses
    VALUES (‘CS’, ‘401’, ‘Operating Systems’, 4.0’);
```

Solution: add the column as invisible

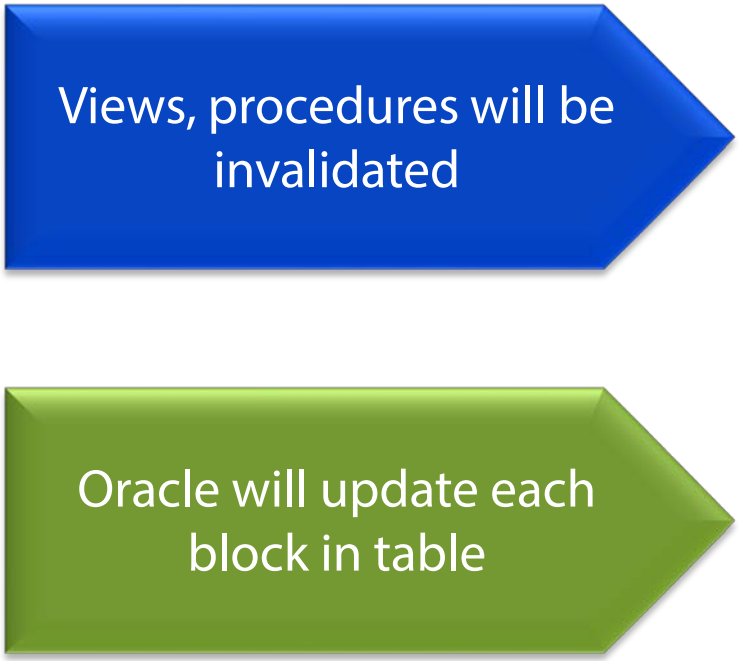
```
ALTER TABLE courses ADD
(
    textbook_name   VARCHAR2(50)  INVISIBLE NULL
);
```

Drop a Column From a Table

Physical Delete of Column

```
ALTER TABLE table_name
  DROP COLUMN (column_name1, column_name2);
```

Physical Delete of Column



Your responsibility to fix these

Can be a performance impact on a large table

Logical Delete of Column

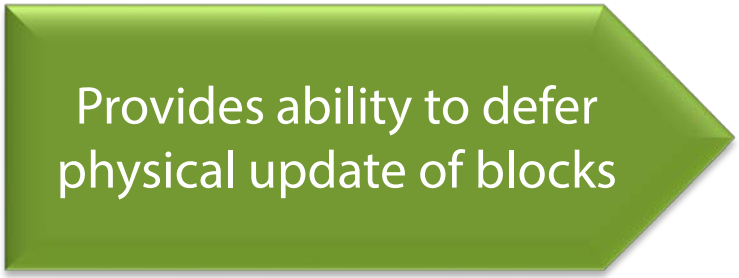
```
ALTER TABLE table_name
    SET UNUSED (column_name1, column_name2);
```

At some later time:

```
ALTER TABLE table_name
    DROP UNUSED COLUMNS;
```

Logical Delete of Column

Regard this as a one way trip



Can occur during off hours or a maintenance window

Dropping Columns and Key Constraints

- Attempt to drop column in a primary or foreign key will result in an error

- To drop the column and the primary/foreign key use

```
ALTER TABLE table_name
    SET UNUSED (column_name1, column_name2)
    CASCADE CONSTRAINTS;
```

- Is dropping a column in a key constraint what you really want?

Changing a Column Data Type

Possible to do, but there are **limitations**

Changing a Column Data Type

Operation	Examples	Allowed?
Increasing size of data type	VARCHAR2(30) → VARCHAR2(50) NUMBER(8,4) → NUMBER(12,6)	Allowed
Decreasing size of VARCHAR	VARCHAR2(50) → VARCHAR2(25)	Allowed if no data truncated
Decrease precision, scale of NUMBER	NUMBER(8,4) → NUMBER(12,6)	Allowed if column empty
Change to different data type	NUMBER(8,4) → VARCHAR2(10) VARCHAR2(10) → DATE	Allowed if column empty

Change Column Data Type Syntax

```
ALTER TABLE table_name MODIFY
(
    column_name      new_data_type
);
```

Module Summary

Oracle Data Dictionary

What is the data dictionary
Useful views

Database Statistics

What are database statistics
How are they used
How are stats gathered

Managing Tables

Dropping tables
Renaming tables

Managing Columns

Adding columns
Renaming columns
Changing column data types

Specialized Table Types

David Berry
http://buildingbettersoftware.blogspot.com/

Module Outline

Temporary Tables

Hold temporary results during a transaction or session

External Tables

Easily import data from flat files

Temporary Tables Overview

- Defined statically
- Table definition available to all sessions
- Data is private to a session
- Act as a private workspace

Basic Create Table Syntax

```
CREATE GLOBAL TEMPORARY TABLE temporary_courses
(
    department_code     VARCHAR2(2)     NOT NULL,
    course_number       NUMBER(3,0)     NOT NULL,
    course_title        VARCHAR2(64)    NOT NULL,
    course_description  VARCHAR2(512)   NOT NULL,
    credits             NUMBER(3,1)     NOT NULL,
    CONSTRAINT pk_courses PRIMARY KEY
        (department_code, course_number),
)
ON COMMIT DELETE ROWS
-- ON COMMIT PRESERVE ROWS;
```

Preserve Rows Options

ON COMMIT DELETE ROWS

Rows automatically deleted when a transaction commits
Default if no option is specified

ON COMMIT PRESERVE ROWS

Rows preserved when a transaction commits
Rows will be deleted when session ends

Additional Rules

Temporary tables cannot participate in foreign key relationships

Indexes can be created on temporary tables

Temporary tables do not have any statistics by default

Database stats are per session starting in Oracle 12c

- **Myles, there will be a demo in here, so it isn't like we are going between two slides in the same format**

- **This slide obviously won't be there either (unless you *really* think I should leave it in.**

External Tables

Common need: importing a flat file

Allow flat file data to be processed by SQL

Data can be filtered, joined or inserted into other tables

External Table Properties

Import only

Only used for **importing** data, not exporting data

File formats

Import both **fixed width** and **delimited** files

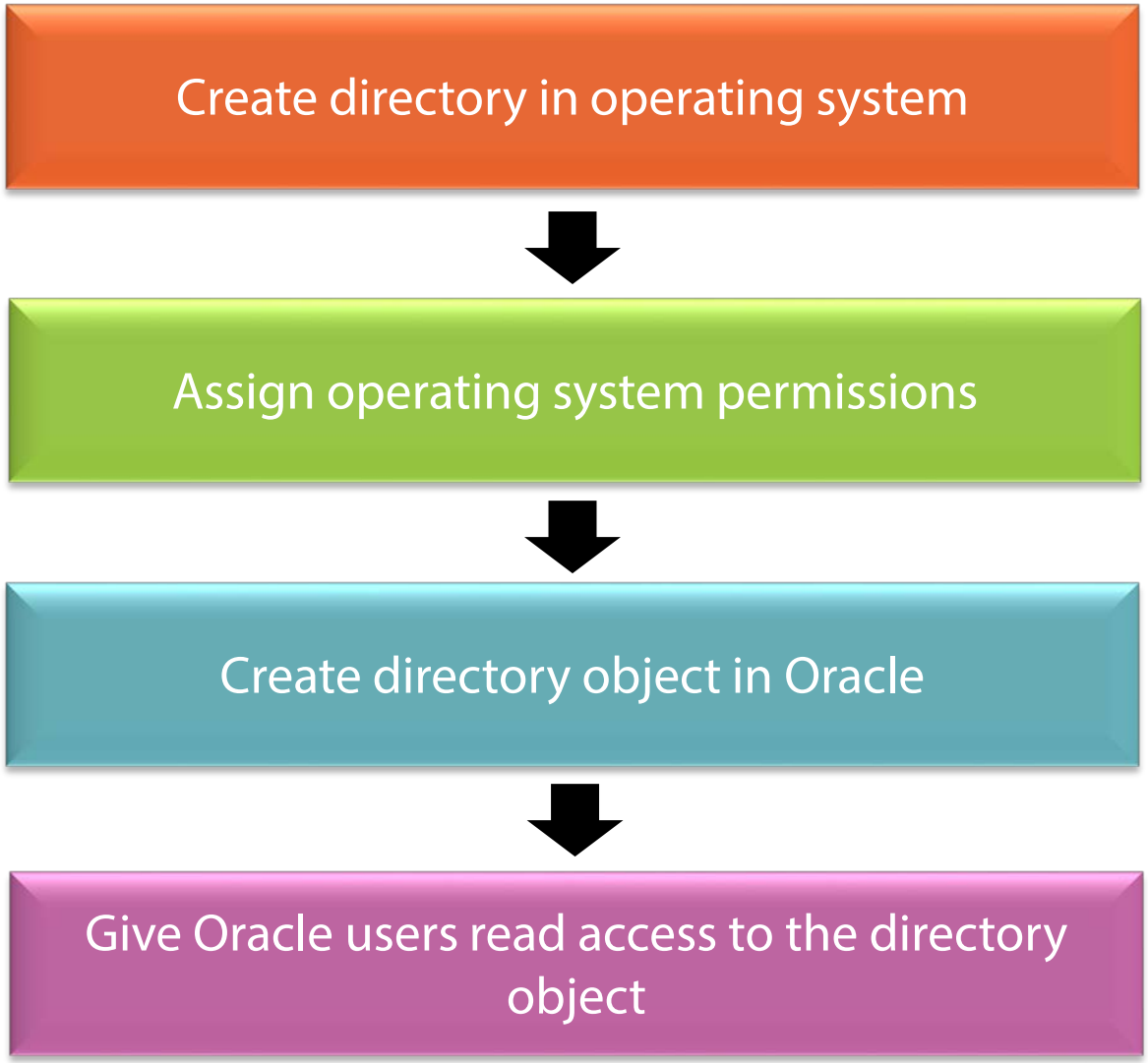
File location

Files must be **located** on the **Oracle server**

Oracle and Directory Objects

- Directory objects tell Oracle where to find data on the server
- Will need system administrator and DBA permissions to set up

Steps to Create a Data Directory



Oracle Directory Objects

```
-- Create the directory object
CREATE DIRECTORY data_import AS
    AS 'C:\data_import\';


-- Grant permissions so a user can use
GRANT READ ON DIRECTORY data_import
    TO student;
```

Create one directory per schema/application

- Protect sensitive data
- Clean separation between applications

External Table Resources

Oracle resources

External Tables Concepts - http://bit.ly/Ora12cExternalTables
Oracle Loader Access Driver - http://bit.ly/OracleLoaderAccessDriver

Sample external table definitions

On my blog - http://bit.ly/SampleExternalTables

When you have errors

Don't forget about the log and discard files

Specialized Tables Summary

Indexes

David Berry
http://buildingbettersoftware.blogspot.com/

Tables

Store data for your application

Indexes

Enable rapid searching of your data

Module Outline

What is an index?

Creating standard (B-tree) indexes

Unique indexes

Function based indexes

Bitmap indexes

Indexing tips

Scanning a Table for Matching Rows

ID	LAST_NAME	FIRST_NAME
213458	Marshall	George
134876	Smith	Ann
142782	Lewis	Gregory
152212	Baker	Charles
108759	Schultz	Sara
254785	Nguyen	Tom
103897	Rodriguez	Carlos
204872	Anderson	Adam
198344	Lin	Jason
175485	Chao	Kiho
154852	Wilson	Jeff

For small tables

- Few rows to search, so data found quickly

For large tables

- Takes a long time – lots of rows to read and process
- Computationally intensive to read and process every row

An index is a **separate data structure** designed to **speed up** the retrieval of rows from a table

The **most common** type of index in Oracle is based is a **B-tree (balanced tree) index**

Index Key

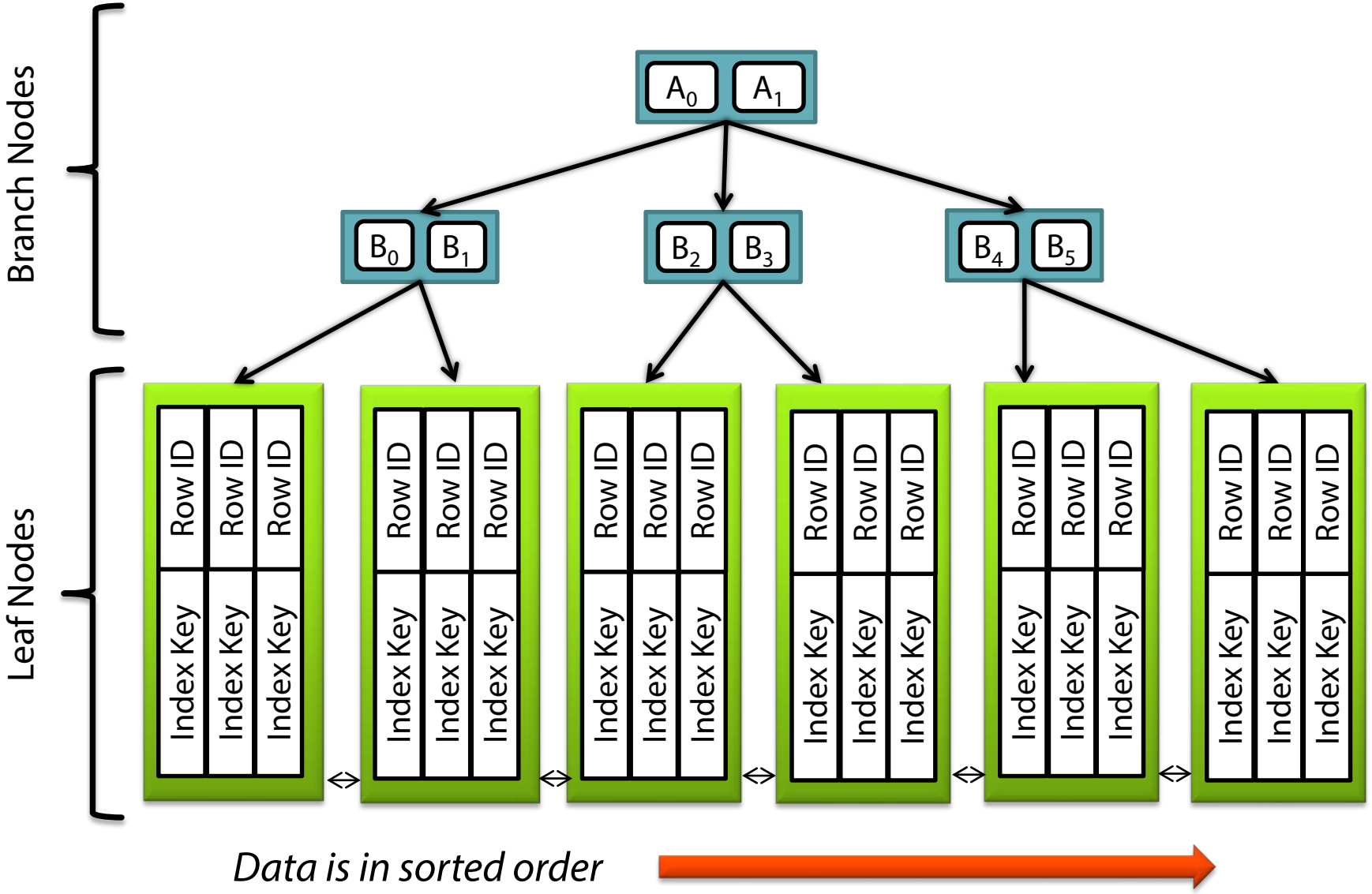
Definition

A subset of columns used to organize and search the index

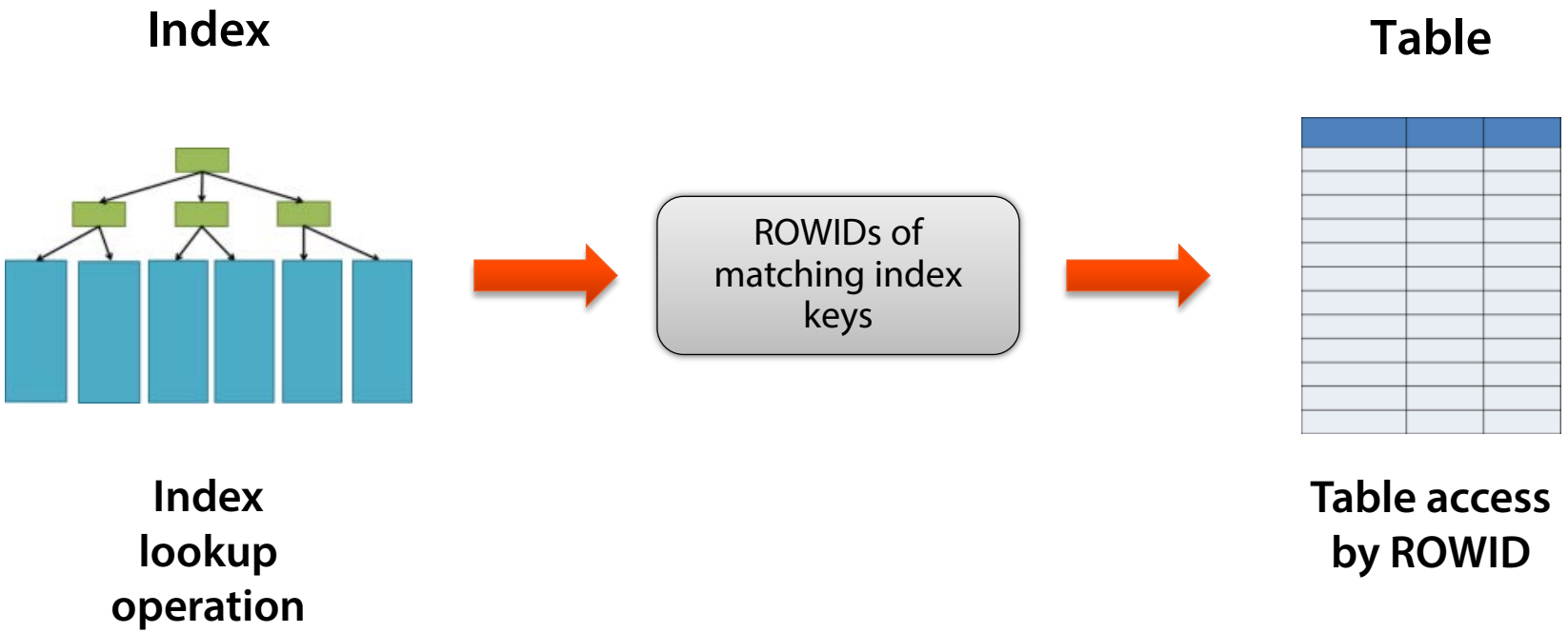
Phone book example

Last name and first name would be the index key

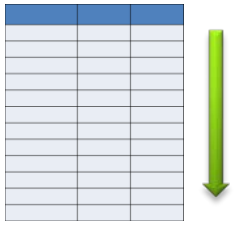
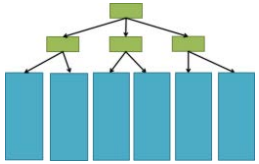
Standard (B-Tree) Index



Using an Index in a SQL Statement



Searching an Index vs. a Full Table Scan

	Table Scan	Index (B-Tree)
		
Search efficiency	O(*n*)	O(log *n*)
Comparisons for searching 1 million items	1,000,000	19

Why Index?

Necessary on larger tables to get **good performance**

SQL statements will **complete faster**

Statements will use **fewer resources** on the Oracle server

Defining an Index

```
CREATE INDEX <<index name>>
    ON <<table name>>
        (<<column 1 name>>, <<column 2 name>> …)
    TABLESPACE <<index tablespace name>>;
```

Defining an Index

```
CREATE INDEX <<index name>>
    ON <<table name>>
        (<<column 1 name>>, <<column 2 name>> …)
    ONLINE
    TABLESPACE <<index tablespace name>>;
```

- Only available in Oracle Enterprise Edition
- Oracle will not lock the table for DML during index creation

Unique Index Introduction

Non-unique indexes

Multiple rows can have the **same value** for the columns in the **index key**

Example: first and last name

Unique indexes

Every row must have a **unique value** for the columns in the **index key**

Example: email address

Defining a Unique Index

```
CREATE UNIQUE INDEX <<index name>>
    ON <<table name>>
        (<<column 1 name>>, <<column 2 name>> …)
    [ONLINE]
    [TABLESPACE <<index tablespace name>>];
```

Function Based Index Introduction

Standard index	Index is created over the **actual column value**
Function based index	Index is created over a **derived value** returned by a function

Classic Example: Case Insensitive Search

```
SELECT *
    FROM applications
    WHERE last_name = 'Thompson'
    AND first_name = 'Eric';
```

What about using the UPPER() function to remove case sensitivity?

```
SELECT *
    FROM applications
    WHERE UPPER(last_name) = UPPER('Thompson')
    AND UPPER(first_name) = UPPER('Eric');
```

PROBLEM! Oracle will no longer use an index on last_name, firsrt_name columns!

Creating a Function Based Index

```
CREATE INDEX fx_applications_fullname
    ON (applications)
    ( UPPER(last_name), UPPER(first_name) );
```

SQL statement with UPPER() function can use this index

```
SELECT *
    FROM applications
    WHERE UPPER(last_name) = UPPER('Thompson')
    AND UPPER(first_name) = UPPER('Eric');
```

Function Based Index Rules

- Can use built in Oracle or user defined functions
- Functions must be deterministic
- Indexes can mix normal column values and derived values from functions

Bitmap Index Introduction

Intended to solve a very **different problem** than a B-tree index

Targeted at columns with **low number** of **distinct values**

Only suitable for **data warehouse** or **read only** applications

Columns With Few Distinct Values

Table data

Customer ID	Gender	Married	Age Range
1001	F	N	18-34
1002	F	Y	35-49
1003	M	Y	35-49
1004	F	Y	35-49
1005	F	Y	18-34
1006	M	Y	62+
1007	F	N	50-61
1008	M	Y	35-49
1009	F	Y	35-49
1010	M	N	35-49
1011	M	Y	35-49
1012	F	N	18-34

Few distinct values in each column

Bitmap Indexes – Simple Example

Table Data

Customer ID	Gender
1001	F
1002	F
1003	M
1004	F
1005	F
1006	M
1007	F
1008	M
1009	F
1010	M
1011	M
1012	F

Bitmap Index

Female	Male
1	0
1	0
0	1
1	0
1	0
0	1
1	0
0	1
1	0
0	1
0	1
1	0

Bitmap Indexes – Many Distinct Values

Table Data

Customer ID	Age Range
1001	18-34
1002	18-34
1003	35-49
1004	35-49
1005	18-34
1006	62+
1007	50-61
1008	35-49
1009	35-49
1010	35-49
1011	35-49
1012	18-34

Bitmap Index

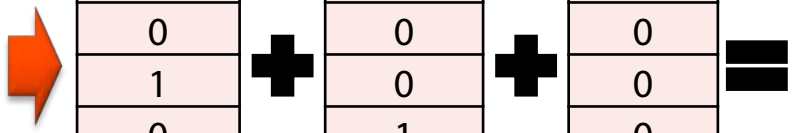
18-34	35-49	50-61	62+
1	0	0	0
1	0	0	0
0	1	0	0
0	1	0	0
1	0	0	0
0	0	0	1
0	0	1	0
0	1	0	0
0	1	0	0
0	1	0	0
0	1	0	0
1	0	0	0

Using Bitmap Indexes

Table data

Customer ID	Gender	Married	Age Range
1001	F	N	18-34
1002	F	Y	35-49
1003	M	Y	35-49
1004	F	Y	35-49
1005	F	Y	18-34
1006	M	Y	62+
1007	F	N	50-61
1008	M	Y	35-49
1009	F	Y	35-49
1010	M	N	35-49
1011	M	Y	35-49
1012	F	N	18-34

Bit arrays from bitmap indexes



Male	Single	35-49	Result
0	1	0	0
0	0	0	0
1	0	1	0
0	0	1	0
0	0	0	0
1	0	0	0
0	1	0	0
1	0	1	0
0	0	1	0
1	1	1	1
1	0	1	0
0	1	0	0

Using Bitmap Indexes

Use for columns with low number of distinct values (few hundred max)

Use only in data warehouse or read only databases, not OLTP databases

Only available in Oracle Enterprise Edition

Defining a Bitmap Index

```
CREATE BITMAP INDEX bx_customers_stats_gender
    ON customer_statistics
    (gender);

// More flexible to create two indexes like this
CREATE BITMAP INDEX bx_customer_stats_age_range
    ON customer_statistics
    (age_range);


// Allowed, not as flexible
CREATE BITMAP INDEX bx_customer_stats_gender_age
    ON customer_statistics
    (gender, age_range);
```

Is there some advice I should know when defining indexes for my database?

Column Order Matters in an Index

Index columns

Leading edge
of the index

- SQL statement must use the first column to use index
- Use as many columns as possible from front of index
- Put columns most frequently used in WHERE clause at the front of the index

Index Selectivity

Moderate selectivity	Better selectivity	Superb selectivity
last_name	last_name first_name	last_name first_name zip_code

More columns or more distinct columns help us identify the data more uniquely

Selectivity Example

Gender column

- Only two distinct values
- Oracle will need to read half the index, then most likely the entire table
- More efficient to use a full table scan (don't bother to read the index)

Email column

- Every row has a unique value
- B-tree structure very effective at efficiently locating unique or nearly unique values
- Data is found reading a minimum number of blocks

Do Not Overindex Your Database

Each index is a separate data structure in Oracle

- Every index takes up space on disk
- Oracle must maintain the index during DML operations against the table

Look at indexes like an investment

- A good index repays it maintenance cost many times over by frequently speeding up other SQL statements
- A poor index is one you pay the cost for but never reap any dividends

Module Summary

What is an index?

How do I create an index?

What is a function based index?

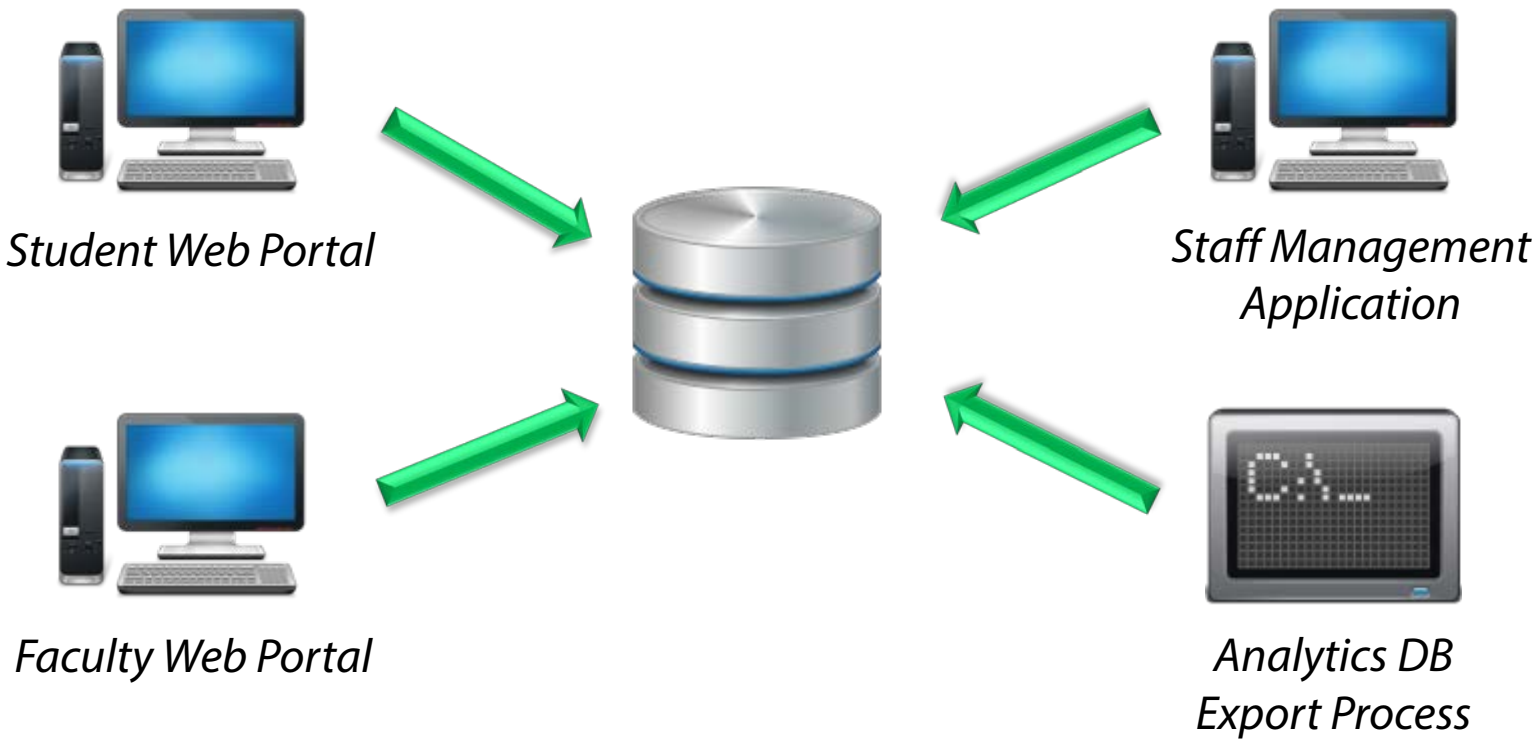
What is a bitmap index?

What are some indexing tips to follow?

Privileges and Roles

David Berry
http://buildingbettersoftware.blogspot.com/

What Applications Use Our Database?

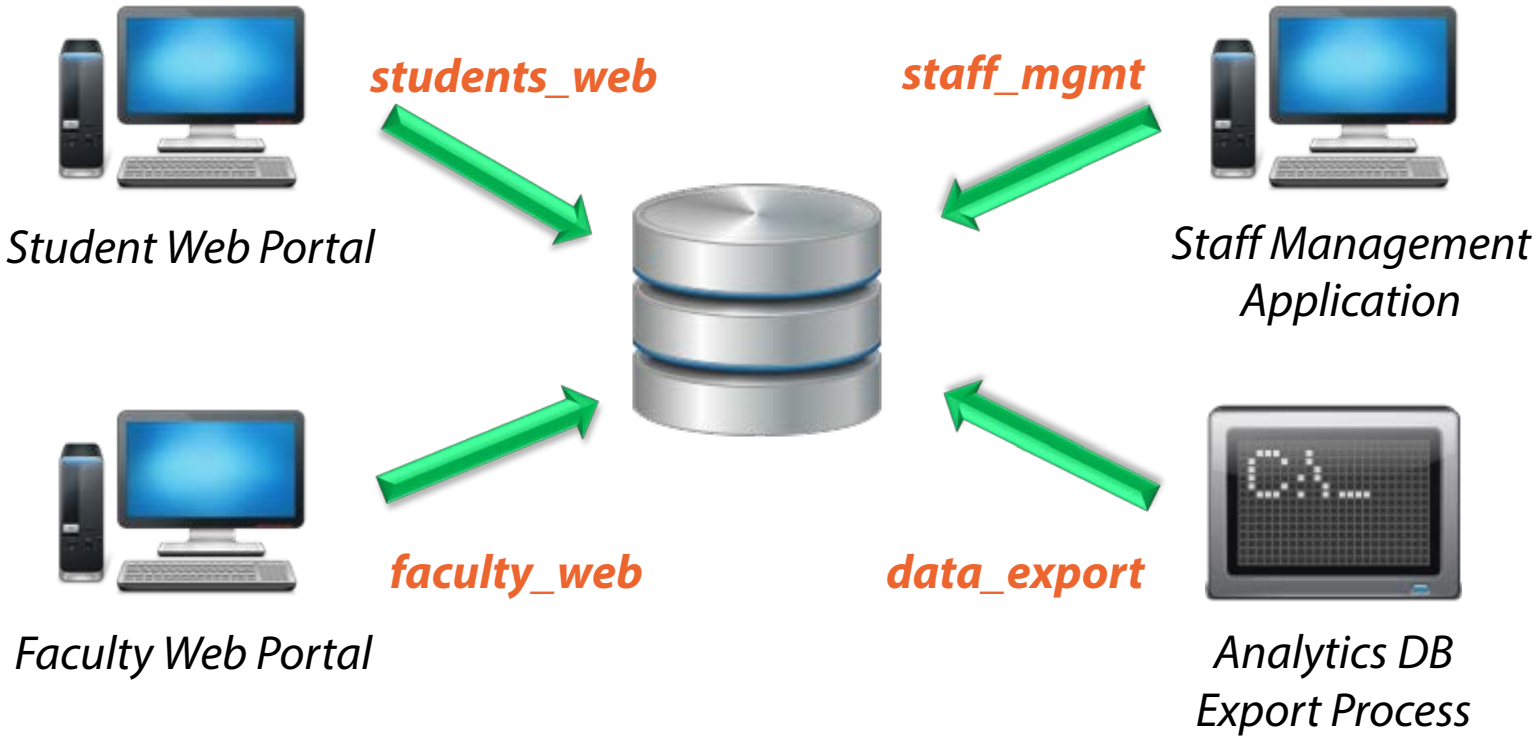


Should all of these applications connect as the user 'student'?

Sharing Users – Problems it Poses

Full Access to All Objects	Application could read, write data it is not supposed to
Schema Changes	Should an application be adding, dropping tables on the fly?
Who is Running SQL	Possible to determine, easier if separate user for each application

What Applications Use Our Database?



Module Outline

GRANT and REVOKE

Object privileges

Column level permissions

Stored procedures and security

Database roles

Security principles

GRANT and REVOKE Commands

Give a user privileges on an object

Remove privileges on an object

GRANT Command

- **Basic syntax**

```
GRANT privileges ON object TO user;
```

- **Examples**

```
GRANT select ON courses TO students_web, faculty_web;


GRANT select, update ON students TO students_web;
```

REVOKE Command

- **Basic syntax**

```
REVOKE privileges ON object FROM user;
```

- **Examples**

```
REVOKE update ON students FROM students_web, faculty_web;

REVOKE insert ON courses FROM faculty_web;
```

Table Privileges

Privilege	Description
ALTER	Change the table definition using ALTER TABLE
DEBUG	Debug the PL/SQL in a trigger on the table
DELETE	Remove rows on the table via a DELETE statement
INDEX	Create an index on the table
INSERT	Add new rows to the table via an INSERT statement
READ	Allows queries from the table but not a SELECT … FOR UPDATE statement (pessimistic locking)
REFERENCES	Allows a foreign key constraint to the table (can only be granted to a user, not a role)
SELECT	Allows reads from the table including SELECT … FOR UPDATE statements
UPDATE	Modify rows in the table via an UPDATE statement

View Privileges

Privilege	Description
DEBUG	Debug the PL/SQL in a trigger on the view
DELETE	Remove rows on the view via a DELETE statement
INSERT	Add new rows to the via via an INSERT statement
READ	Allows queries from the view but not a SELECT … FOR UPDATE statement (pessimistic locking)
REFERENCES	Allows a foreign key constraint to the view (can only be granted to a user, not a role)
SELECT	Allows reads from the view including SELECT … FOR UPDATE statements
UPDATE	Modify rows in the view via an UPDATE statement

Column Level Privileges

```
GRANT select, update ON students TO students_web;
```

*User students_web can **update any column** on the students table*

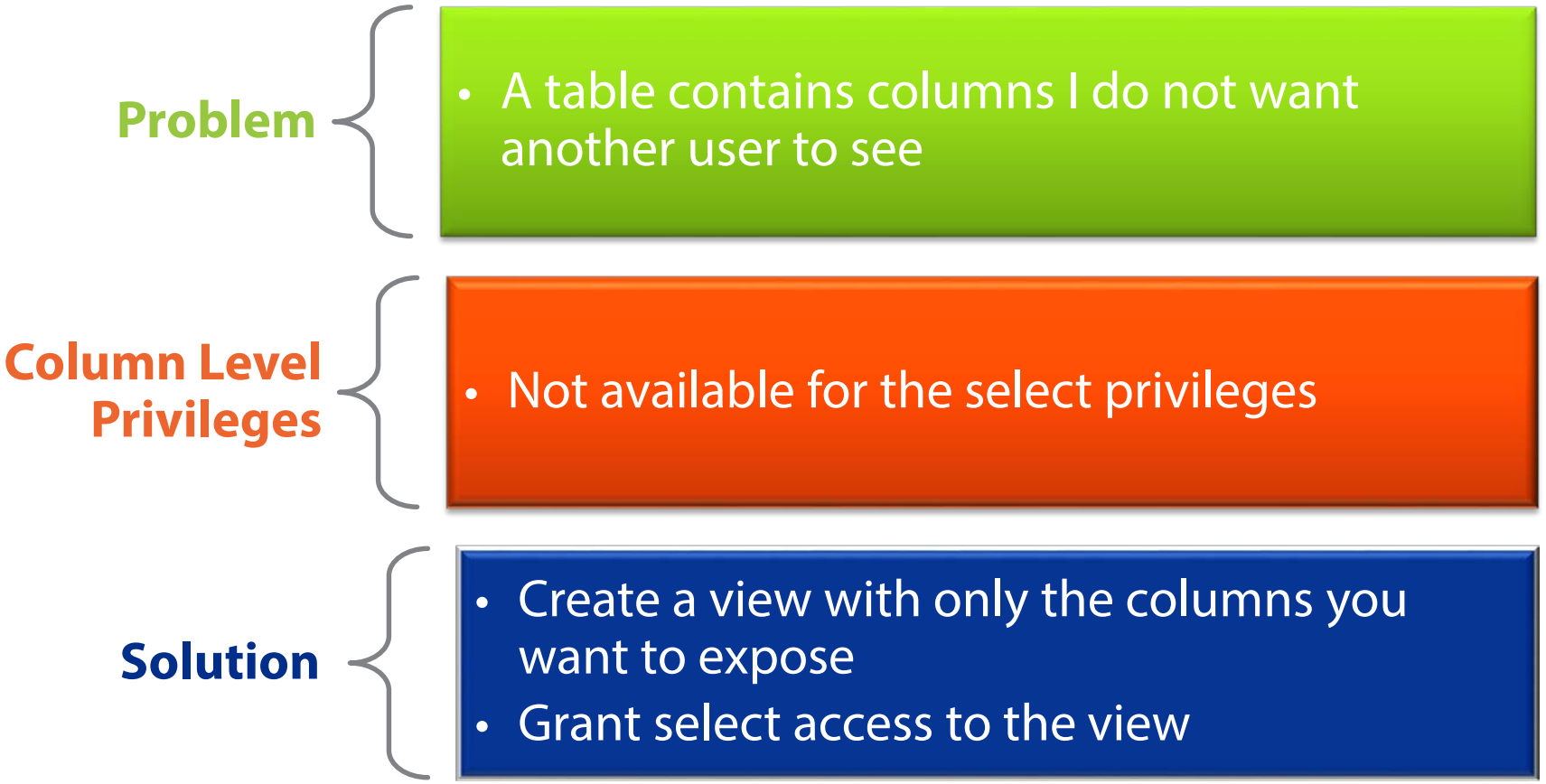
Column Level Privileges

```
GRANT update (street_address, city, state, telephone, email)
    ON students TO students_web;
```

User students_web can only update columns ***included in this list***

- Individual columns can be supplied for the insert, update and references privileges

Using a View to Restrict Visible Columns



Example – Employee Phone Directory

```
desc employees;

Name             Null     Type
---------------- -------- ------------
EMPLOYEE_ID      NOT NULL  NUMBER(8),
FIRST_NAME       NOT NULL  VARCHAR2(30),
LAST_NAME        NOT NULL  VARCHAR2(30),
DATE_OF_BIRTH    NOT NULL  DATE,
TITLE            NOT NULL  VARCHAR2(30),
DEPARTMENT_ID    NOT NULL  NUMBER(4),
PHONE            NOT NULL  VARCHAR2(20),
EMAIL            NOT NULL  VARCHAR2(60),
SALARY           NOT NULL  NUMBER(10,2)
```

Example – Employee Phone Directory

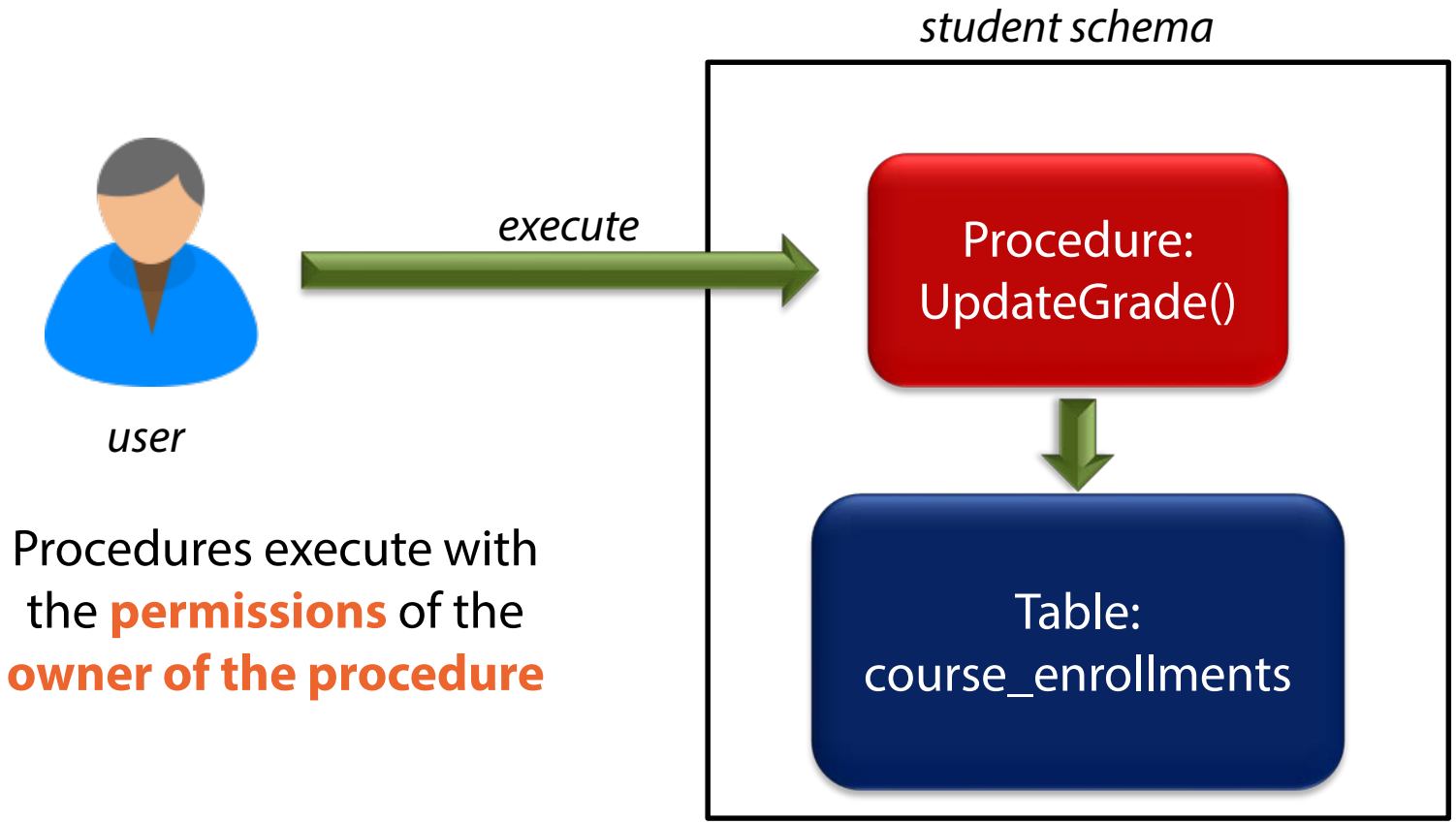
```
CREATE VIEW v_employee_contact_info AS
    SELECT
        employee_id, first_name, last_name,
        title, department_id, phone, email
    FROM employees;


GRANT select ON v_employee_contact_info TO web_user;
```

Stored Procedure/Function Privileges

Access to run the procedure or function

Access to debug through the procedure/function

Procedure Execution Permissions



Procedures execute with the **permissions** of the **owner of the procedure**

SQL in Applications

Versus

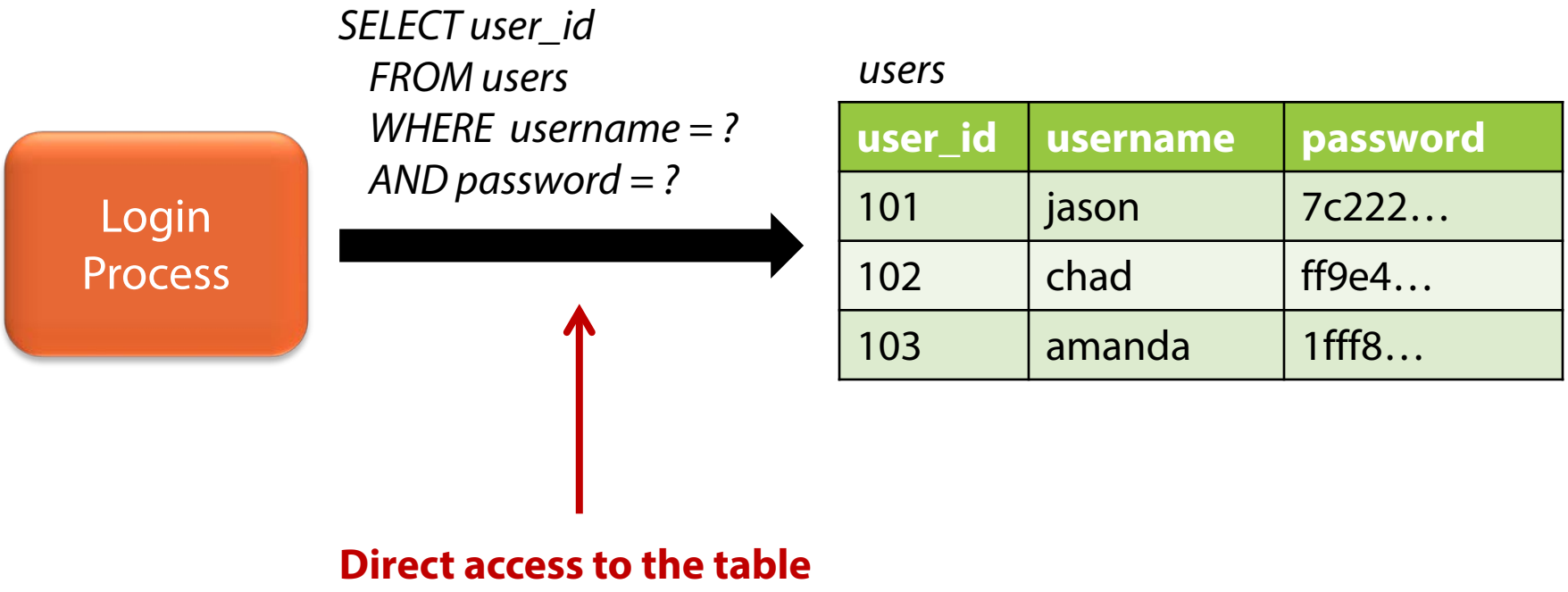
Stored Procedures

Debated numerous times

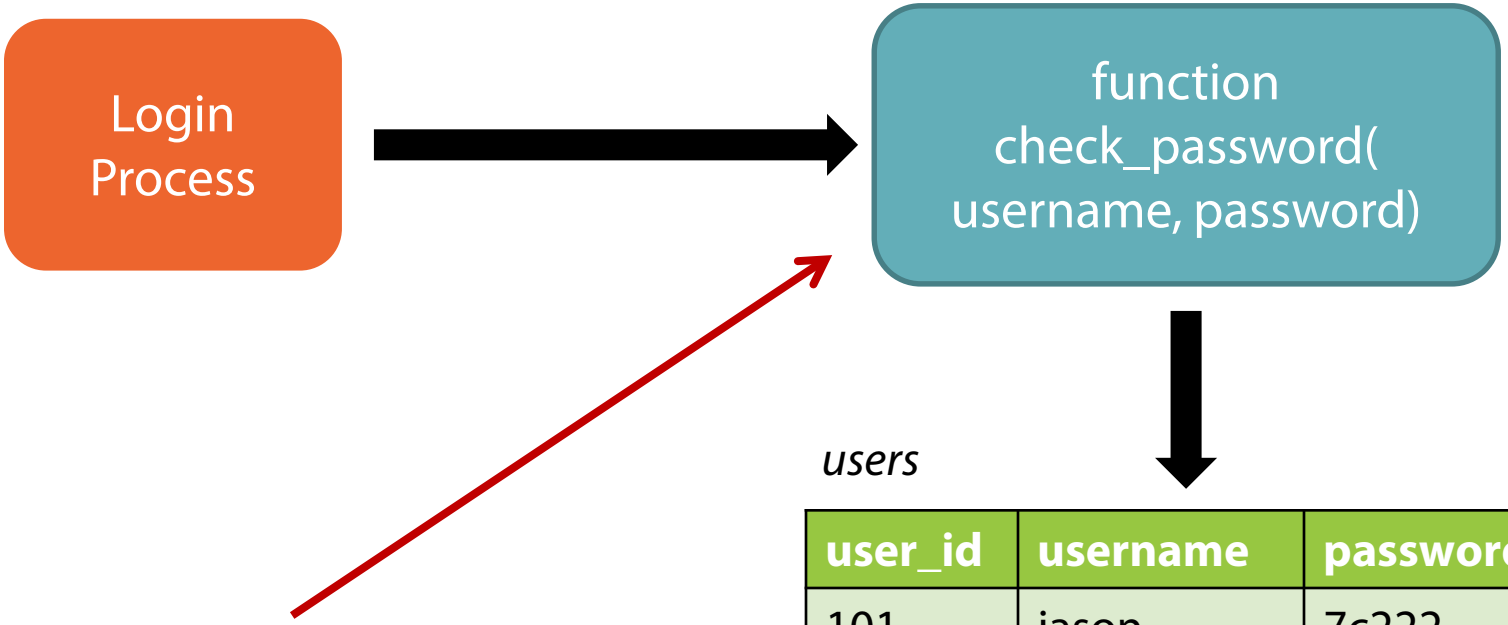
No need to debate again here

Procedures can offer security benefits

Direct Table Access and Security Concerns



Stored Procedures and Security



user_id	username	password
101	jason	7c222…
102	chad	ff9e4…
103	amanda	1fff8…

Stored Procedure Wrap Up

Grant **EXECUTE** privilege to allow users to run

Users only need EXECUTE privilege, **not** access to **underlying objects**

Can **restrict access** to a specific use case

Consider using when you have **sensitive data**

Roles Introduction

```
GRANT select ON departments TO john;
GRANT select ON courses TO john;
GRANT select ON students TO john;
…

GRANT select ON departments TO erica;
GRANT select ON courses TO erica;
GRANT select ON students TO erica;
…

GRANT select ON departments TO robert;
GRANT select ON courses TO robert;
GRANT select ON students TO robert;
…
```

Database Roles

Definition

Container to group a set of privileges together

Benefit

Simplify management of permissions

Role Management

```
-- Create the role
-- Most likely will need to be done by a DBA
CREATE ROLE technology_staff;

-- Assign privileges to role
GRANT select ON departments TO technology_staff;
GRANT select ON courses TO technology_staff;
GRANT select ON students TO technology_staff;
…

-- Assign the role to users
-- Again, performed by a DBA
GRANT technology_staff TO john;
GRANT technology_staff TO erica;
GRANT technology_staff TO robert;
```

Thoughts on Database Privileges

Principle of Least Privilege

Only grant access to the data that an application or user absolutely needs and no more

Defense in Depth

Use multiple mechanisms to secure access to your data

Some Good Practices

Separate application users from the schema owner

Schema owner has full access to all objects
Schema owner can perform DDL statements

Use a separate Oracle user for each application

Applications usually have different access requirements
Easier to tell what application a statement was executed from

Take time to assign appropriate access permissions

Don't rely solely on application code to enforce security
Only give a user the access they need

Summary

- Multiple users
- GRANT and REVOKE
- Object privileges
- Column level permissions
- Stored procedures and security
- Database roles
- Security principles

Good Practices

David Berry
http://buildingbettersoftware.blogspot.com/

Module Overview

- Naming standards and conventions
- Table and column comments
- Single responsibility principle

Naming Standards and Conventions

Establish and use standards for your organization

Consistent use makes your database easier to work with

Standards add an element of professionalism

Naming Tables and Columns

Use a descriptive name of what data is contained in the table or column

Table names are commonly nouns

Be consistent in use of singular or plural forms of names

Column names should be specific about what attribute they represent

Favor Full Words Over Abbreviations

Full words are **more descriptive**

Abbreviations are open to **misinterpretation**

Common abbreviations in **one culture** may be **confusing to others**

User Underscores to Improve Readability

Which set of names is easier to read?

STREET_ADDRESS	STREETADDRESS
DEPARTMENT_NAME	DEPARTMENTNAME
REQUEST_ID	REQUESTID
COURSE_TITLE	COURSETITLE

Name Your Table Constraints

Constraint type	Naming convention
Primary Key	PK_<table name>
Foreign Key	FK_<parent table name>_<parent column>
Check Constraint	CK_<table name>_<column name>

Naming Indexes

<prefix>_<table name>_<column name(s)>

Based on index type

IX	Non-unique index
UX	Unique index
FX	Function based index

Goal: Look at an index and be able to tell its purpose

Most Important Rule for Naming Objects

Be expressive

Database Comments

Needed when a concept is not self explanatory

Documentation works best when it can be stored with the structure it is documenting

Adding Comments to Tables and Columns

```
-- Setting comments on a table
COMMENT ON TABLE <<table name>> IS ‘<<comment test>>’;


-- Setting a comment on a columns
COMMENT ON COLUMN <<table name>>.<<column name>>
    IS ‘<<comment text>>’;
```

Selecting Comments About Tables and Columns

```
SELECT * FROM user_tab_comments;

SELECT * FROM user_col_comments
    WHERE table_name = '<<table name>>';


SELECT tc.column_name, tc.data_type, tc.data_length,
    tc.data_precision, tc.data_scale, tc.nullable,
    cc.comments
FROM user_tab_columns tc
INNER JOIN user_col_comments cc
    ON tc.table_name = cc.table_name
    AND tc.column_name = cc.column_name
WHERE tc.table_name = '<<table name>>'
ORDER BY tc.column_id;
```

Single Responsibility Principle

Every structure in your database
should do exactly one thing

Tables should store a single type of entity

Do Some Columns Apply Only for Some Rows?

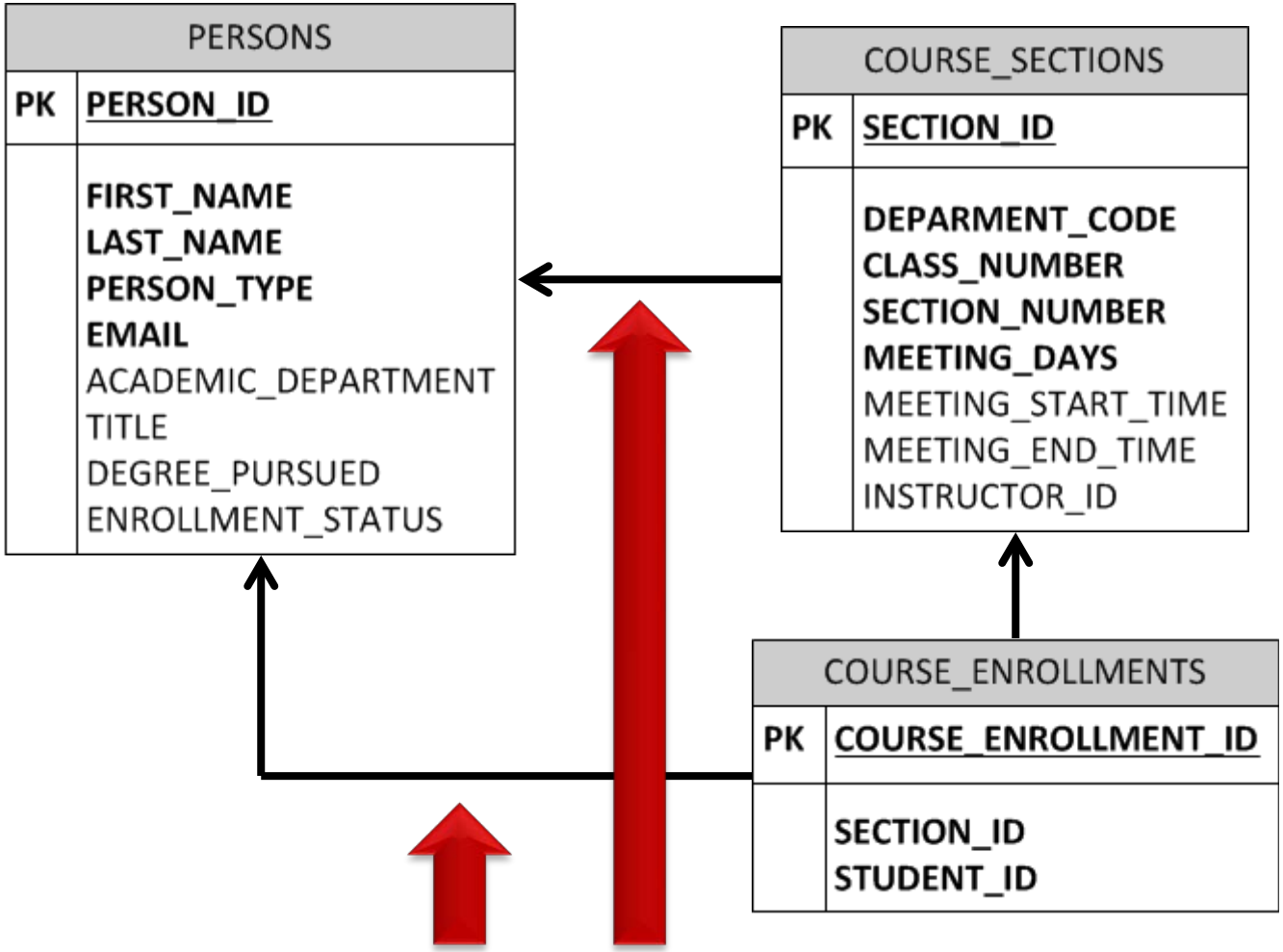
PERSON ID	FIRST NAME	LAST NAME	PERSON TYPE	EMAIL	ACADEMIC DEPT	TITLE	MAJOR	STATUS
1001	John	Randolph	S	jrandolph@...			CS	G
1002	Erica	Schmidt	S	eschmidt@...			EE	G
1003	Richard	Rodgers	S	rrodgers@...			Math	W
1004	Jennifer	Meehan	F	jmeehan@...	Math	Professor		
1005	Martha	McNeil	F	mmcneil@...	Physics	Assoc Professor		
1006	Eric	O'Reilly	S	eoreilly@...			EE	E
1007	William	Anderson	F	wanderson@...	Chemistry	Professor		
1008	Matthew	Taylor	S	mtaylor@...			ME	E

Attributes apply to all rows

Only for faculty members

Only for students

Foreign Key Issues



Problem: Cannot create foreign key to only faculty or only student records

Columns Should Have One Use

ORDER_ID	SHIP_STATUS	SHIP_DATE
1001	S	2014-07-15
1002	S	2014-07-17
1003	S	2014-07-18
1004	P	2014-07-25
1005	P	2014-07-27

Actual ship dates

Expected ship dates

A Better Design

ORDER_ID	SHIP_STATUS	TARGET_SHIP_DATE	ACTUAL_SHIP_DATE
1001	S	2014-07-15	2014-07-15
1002	S	2014-07-18	2014-07-17
1003	S	2014-07-19	2014-07-18
1004	P	2014-07-25	
1005	P	2014-07-27	

Single Responsibility Principle Benefits

Easier to understand the design

Easier to access data in tables

Easier to achieve good design

Summary


Naming conventions
Database comments
Single responsibility principle